Robust and Efficient Uncertainty-aware Multi-Object Tracking Through Vision-Based Ego-Motion Awareness

by

Mohammad Jani
B.Sc., Iran University of Science and Technology, 2021

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Robust and Efficient Uncertainty-aware Multi-Object Tracking Through Vision-Based Ego-Motion Awareness

by

Mohammad Jani

B.Sc., Iran University of Science and Technology, 2021

Supervisory Committee

Dr. H. Najjaran, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Yang Shi, Supervisory Committee Member

(Department of Mechanical Engineering)

ABSTRACT

This thesis presents the development and refinement of the UVEMAP system, an **U**ncertainty-aware **V**ision-based **E**go-**M**otion-**A**ware target **P**rediction module designed for robust multi-object tracking using only monocular camera inputs. This work aims at improving the performance of Multi-Object Tracking (MOT) systems by the incorporation of ego-motion and depth estimation uncertainty through a heuristic and computationally efficient solution.

The first step involves separating the impact of camera movement in Kalman-based MOT algorithms. This helped to increase tracking precision by ensuring that the vehicle ego-motion did not negatively impact the targets' anticipated location. The work is continued by creating a pure vision system that uses only the image stream from a monocular camera; thus removing the requirement for other sensor data, like IMUs, GPS, and wheel encoders. This strategy improves the system adaptability and applicability in many contexts as the proposed solution can completely be agnostic of the metric scale such as depth and translation vector.

One of the main contributions is the integration of visual odometry and depth estimation using a modified Monodepth2, which estimates depth and camera motion by employing a self-supervisory signal generated by image reprojection error. The modifications to Monodepth2 guarantee its compatibility with UVEMAP, allowing precise depth and pose estimation from monocular images. In order to account for uncertainty in depth estimation, a conformal prediction method is applied which identifies prediction intervals to gauge the level of uncertainty associated with each depth estimation by computing nonconformity scores in the dataset. This utilization of data enhanced the capability of the Kalman filter to handle occlusions and noisy readings.

The incorporation of unified scale depth and pose estimations as well as depth

uncertainty quantification into the proposed target prediction module resulted in a substantial improvement in performance metrics compared to baseline methods. Experiments conducted on the KITTI dataset show that UVEMAP significantly reduces identity switches and enhances tracking accuracy and robustness. The computational efficiency of the proposed method, stemming from its heuristic nature, makes it suitable for deployment on edge devices, including autonomous ground robots and vehicles. This research makes a notable contribution to the field of multi-object tracking by presenting a comprehensive framework that integrates ego-motion awareness and uncertainty quantification, all from a monocular video stream, to achieve superior tracking performance.

Table of Contents

List of Tables

List of Figures

Acronyms

AssA Association Accuracy.

AssR Association Recall.

AW Adaptive Weighting.

CMC Camera Motion Compensation.

CNN Convolutional Neural Network.

DA Dynamic Appearance.

DBT Detection-Based Tracking.

ECC Enhanced Correlation Coefficient.

EKF Extended Kalman Filter.

EMAP Ego-Motion Aware Target Prediction.

FN False Negatives.

FP False Positives.

FPS Frames Per Second.

HOTA Higher Order Tracking Accuracy.

IDF1 ID F1 Score.

IoU Intersection over Union.

KF Kalman Filter.

MOT Multi-Object Tracking.

MOTA Multi-Object Tracking Accuracy.

OC-SORT Observation-Centric SORT.

OCM Observation-Centric Momentum.

ORU Observation-centric Re-Update.

Re-ID Re-Identification.

RNN Recurrent Neural Network.

ROS Robot Operating System.

SLAM Simultaneous Localization And Mapping.

SORT Simple Online and Realtime Tracking.

VO Visual Odometry.

Nomenclature

$\mathbf{I}_n$ image frame at time step n

(s, r) area and ratio of the object bounding box

(u, v) horizontal and vertical pixel location of the center of an object

(u^l, v^t) top-left corner of a bounding box

(u^r, v^b) bottom-right corner of a bounding box

$(\dot{x}^l, \dot{y}^t, \dot{x}^r, \dot{y}^b)$ horizontal and vertical pixel velocities of the corners of the target object bounding box caused solely by the object's motion in the world frame

α miscoverage rate for conformal prediction

γ angle between the center of the object and camera principal axis

$\hat{\mu}(X)$ trainable function of machine learning model parameters

$\mathbf{b}_n^j$ bounding box of object j in frame n

$\mathbf{d}_n^j$ detection of object j in frame n

$\mathbf{F}$ state transition matrix

$\mathbf{G}$ input transition matrix

$\mathbf{H}$	observation matrix
$\mathbf{P}$	state covariance matrix
$\mathbf{Q}$	process noise covariance matrix
$\mathbf{R}$	measurement noise covariance matrix
$\mathbf{u}$	input vector
$\mathbf{v}$	process noise vector
$\mathbf{w}$	measurement noise vector
$\mathbf{x}$	state definition of object
$\mathbf{x}_n^i$	state definition of object i in frame n
$\mathbf{z}$	observation vector
$\mathcal{D}_n$	set of detections in frame n
$\mathcal{M}$	sample machine learning model
$\mathcal{S}_1$	training set
$\mathcal{S}_2$	calibration set
$\mathcal{T}$	set of all active trajectories
$\mathcal{T}_i$	track sequence of object states over time
ψ	y-axis rotation of camera
A	association matrix
$C(X)$	prediction interval for the sample X

c_n^j	class label of object
D	ego-motion displacement vector
d	depth of the object
f	camera focal length
P_{XY}	joint probability distribution of X and Y
$Q_{1-\alpha}$	$(1-\alpha) - thquantileofthecalibrationscores$
r_i	absolute residual score
s_i	nonconformity score
s_n^j	confidence score of detection
X, Y	training or testing sample pair

PREFACE

Chapter 1 includes material that has been published on *arXiv* as a paper preprint [1] and has been submitted to a journal for possible publication. This material includes a review on object detection and neural network compression methods. Chapter 1 and Chapter 4 contain concepts and materials from another paper published on *arXiv* as an equal-contribution paper [2]. These include an introduction to multi-object tracking in Chapter 1 and camera motion integration in Chapter 4.

In the first-mentioned work, I, Mohammad Jani, was responsible for conceptualization, data curation, original draft preparation, and writing and editing the manuscripts. Jamil Fayyad and Younes Al-Younes contributed to the conceptualization and reviewing of the drafts.

In the second-mentioned work, I contributed to algorithm development, especially in camera motion extraction and Kalman Filter augmentation, code implementation, visualization, and manuscript writing. Navid Mahdian contributed to the structural design of the method, code implementation, dataset visualization, and manuscript writing.

ACKNOWLEDGEMENTS

I am sincerely thankful to my family and friends for their unswerving support, and Dr. Homayoun Najjaran for his mentoring, patience, and trust in my work throughout my master's program. I am also grateful to my amazing teammate Navid Mahdian for his invaluable contributions to this research. I extend my gratitude to Sentire Co. for their guidance and resources that enabled this research to be done smoothly. Lastly, I would like to thank my dear friend Yashar Rahimi for his priceless support and for being there for me in my low moments.

Chapter 1

Introduction

Multi-object tracking (MOT) is a fundamental task in computer vision that involves detecting and tracking multiple objects across a sequence of video frames. This complex problem is critical for a wide range of applications, including autonomous driving, where it ensures the safe navigation of vehicles; surveillance, which relies on accurate tracking for security and monitoring purposes; sports analysis, where it helps in tracking players and the ball to provide detailed statistics and insights; and robotics, where it is essential for interaction and navigation in dynamic environments. In this chapter, the motivation behind studying MOT will be discussed; outlining the objectives of this research, and providing an overview of the subsequent chapters.

1.1 Motivation

Multi-Object Tracking (MOT) plays a significant role in computer vision, with various applications that significantly impact our daily lives and numerous industries. Primarily, this task involves identifying and tracking multiple objects across a sequence of images, which presents challenges such as occlusions, alternation in object appearance, and changes in object velocities. This technology is pivotal in many contexts,

such as autonomous vehicle navigation, where precise tracking of other vehicles on the road and pedestrians is necessary for the ego vehicle to operate safely. It also plays a central role in surveillance systems, enabling the surveillance of pedestrians in densely populated areas for security purposes. Additionally, multi-object tracking is instrumental in sports analysis as it provides detailed tracking of players and objects, which offers rich and valuable insights into game dynamics. Furthermore, in robotics applications, it facilitates the interaction between robots and their environment, allowing for object avoidance, navigation, and environment mapping. Ego-motion awareness is a crucial aspect of enhancing the performance of multi-object tracking algorithms, mainly because it enables tracking systems to comprehend and account for their own motion within the environment. By incorporating ego-motion awareness, trackers can distinguish the impact of ego-motion on the motion of targets within the image frame. Such integration increases tracking accuracy and robustness; for example, in autonomous vehicle navigation, understanding the motion of the vehicle allows for more precise tracking of surrounding objects by compensating for the movements of the ego vehicle. This leads to safer navigation decisions as the system can accurately predict the trajectories of other objects.

Having ego-motion awareness as a modular component within multi-object tracking systems offers multiple benefits. Firstly, it can be integrated with various tracking algorithms and enhance performance without requiring extensive modifications. Secondly, it allows the ego-motion-aware prediction component to be modified and improved independently from the tracking system. Lastly, this isolation helps developers focus on optimizing this aspect for different sensor setups and application needs, further enhancing the versatility and applicability of multi-object tracking algorithms in different domains.

This thesis is a continuation of a previous joint project conducted in the Advanced

Control and Intelligent Systems (ACIS) Lab. The basic proposed solution, EMAP, was developed in a joint research with equal collaboration. Building upon this work, I, Mohammad Jani, developed VEMAP and UVEMAP, which rely solely on monocular camera video streams. The previous solution depended on odometry and depth map data, which limited the applicability of the method. By focusing on a pure vision-based approach, this research aims to overcome these limitations and enhance the versatility and deployment potential of the system.

Extracting ego-motion simply can be done through sensory data such as wheel encoders, GPS, and IMUs. However, in cases where such data is not available, the effectiveness of naive ego-motion awareness is hindered. Therefore, focusing on a pure vision-based solution is beneficial as it requires no other source data except the image stream.

Ego-motion and depth estimation from mono camera naturally comes with a wide variety of uncertainties caused by different sources such as image noise. Having a reliable MOT solution that predicts object location without absurd assumptions about the uncertainties is desirable. It accurately models the uncertainty and utilizes it to predict not only object location but also its confidence accurately.

1.2 Objectives

The main objectives of this research are as follows:

1. **Develop an Enhanced Multi-Object Tracking Framework:** Create a robust MOT system that integrates ego-motion awareness to improve tracking accuracy and reliability in dynamic environments.

2. **Incorporate Pure Vision-Based Ego-Motion Estimation:** Implement a pure vision-based ego-motion estimation module to eliminate the dependency

on additional sensory data, making the system more versatile and cost-effective.

3. **Quantify and Utilize Uncertainty in Tracking:** Introduce conformal prediction techniques to quantify the uncertainty in object tracking and incorporate this information into the tracking process to enhance the system's robustness against noisy measurements and occlusions.

4. **Evaluate the System in different Scenarios:** Conduct extensive experiments to evaluate the performance of the proposed MOT system in various scenarios in autonomous driving.

5. **Compare with State-of-the-Art Methods:** Benchmark the proposed system against existing state-of-the-art MOT algorithms to demonstrate its superiority in terms of accuracy, robustness, and computational efficiency.

By achieving these objectives, this research aims to contribute significantly to the field of multi-object tracking, providing a robust and efficient solution for various real-world applications.

1.3 Thesis Organization

This thesis contains several chapters which detail different views of robust and efficient multi-object tracking. Provided below is the outline of this work:

Chapter 2 explains the required background knowledge of the different submodules of the work and provides related work respectively.

Chapter 3 describes in detail the MOT problem and represents its mathematical foundation.

Chapter 4 details how the new solution for multi-object tracking works by decoupling the camera motion effect and integrating it into the Kalman Filter.

Chapter 5 explains the experiment selected to evaluate the new method and provides the corresponding results along with an in-depth analysis.

Chapter 6 contains the summary of how the new solution is impactful and discusses the limitation of the method which paves the way for future work.

Chapter 2

Background and Related Work

2.1 Multi-Object Tracking

Multi-Object Tracking (MOT) is a pivotal task in computer vision, with applications spanning various domains such as monitoring crowds in surveillance systems, analyzing player movements in sports, tracking vehicles on roads, and observing marine life in oceans [3].

Detection-Based Tracking (DBT) is commonly utilized in various applications because it efficiently handles situations where objects may enter or exit the scene. In the DBT approach, multi-object tracking (MOT) is conceptualized as a graph optimization challenge. In this model, an object detector identifies and encloses objects within bounding boxes in each frame, which act as the nodes of a graph. Connections between these nodes are established if they represent the same object across frames. The primary constraints for this graph-based problem are: 1) no edges are allowed to point backward in time, 2) each node can be linked to at most one other node, and 3) each node can only have one incoming connection.

Consequently, the MOT problem simplifies to finding the best associations be-

tween nodes in the graph. Methods like Tracktor++ [4] aim to optimize these associations, assuming high frame rates and continuous object detections. However, this assumption is often unrealistic in real-world situations where factors like camera movement and occlusions occur. When detections are not consistently available, it is necessary to either discard the frame-to-frame association assumption or replace missing detections with predicted object states.

Most of the leading MOT methods [5–10] adhere to the detection-based tracking approach, which involves three sequential tasks: detection, prediction, and association. Recent advancements in object detectors [11–13] have allowed DBT techniques to concentrate on refining the association process between detection bounding boxes in each frame.

The effectiveness of these techniques largely depends on the accuracy of the detection module. When the detector fails to identify target objects correctly, the prediction module must estimate the location of the missed object until it is either detected again or its track is terminated. The prediction module aims to forecast the object's location in future frames based on its previous trajectory. Many prominent methods [7–9] employ the Kalman filter as their prediction model. In [5], the authors used a Kalman Filter with a constant velocity motion model to predict future object positions. OC-SORT [7] built upon this by introducing a technique to mitigate noise accumulation during occlusions, creating a virtual trajectory for the occlusion period while preserving the simplicity of the constant velocity Kalman Filter model.

Some methods [14, 15] treat the prediction and association phases as a combined problem and utilize Recurrent Neural Networks (RNNs) to address it. For example, the authors of [14] introduced an RNN framework that incorporates motion, appearance, and interaction features to perform prediction and association over a temporal window. Other research efforts [9, 16, 17] have focused on enhancing the association

phase by leveraging appearance features. In [9], the authors developed a deep association metric by training a Convolutional Neural Network (CNN) on the MARS [18] person re-identification dataset, thereby improving the accuracy and robustness of the association process.

In [16], the visual similarity between intra-frame target bounding boxes is computed using a Gaussian mixture probability hypothesis density filter [19] allowing for effective tracking of multiple targets across varying densities. Additionally, [17] introduced a novel unsupervised re-identification learning module, which removes the requirement for object labels. This approach includes an occlusion estimation module to predict occluded regions, achieving performance comparable to that of supervised methods.

2.2 Object Detection

As a fundamental problem, object detection has been an active area of research for many years. The main goal of object detection involves identifying and localizing objects of interest from different categories within some given image. Object detection is the basis of many other advanced computer vision tasks [20] ranging from semantic segmentation [21] to object tracking [22] to activity recognition [23]. In recent years, deep learning-based approaches such as Convolutional Neural Networks (CNNs) have achieved state-of-the-art performance in object detection tasks. As a result of the advancements in computational power and cutting-edge algorithms, object detection has become more accurate, enabling a wide range of real-world applications. Compared to classical object detection methods, using CNNs alleviates the problem of feature extraction, classification, and localization in object detection [24–29]. Regarding this, object detection can be done through two methods, namely, single-stage

and two-stage detection. While in the former, the algorithm directly predicts the bounding boxes and class probabilities for objects, in the latter, the algorithm first generates a set of region proposals and then classifies those proposals as objects or backgrounds [30]. Unlike Faster R-CNN [25] and R-FCN [31] as two-stage object detection methods, single-stage ones, like YOLO [24], SSD [32], EfficientDet [12], and RetinaNet [13], typically use one Fully Convolutional Neural Network (FCN) to detect objects' classes and spatial locations.

Among different single-stage object detection methods, YOLO has gained a lot of attention since it was published in 2016. The primary idea behind YOLO is to divide an input image into a grid of cells and predict bounding boxes and class probabilities for each cell. YOLO treats object detection as a regression problem. Also, since it uses a single neural network for object detection and classification, it can be optimized jointly for both tasks, which ultimately enhances the whole detection performance. YOLOv1 was equipped with a simple structure containing 24 convolutional layers and two fully connected layers at the end to deliver probabilities and coordinates [24]. Since its introduction, YOLO has undergone several improvements and variations. In 2017, YOLOv2 (also known as YOLO9000) was published with improvements in the performance led by using multi-scale training, anchor boxes, batch normalization, Darknet-19 architecture, and a modified loss function [33]. Following that, Redmon and Farhadi introduced YOLOv3 that employs a feature pyramid network, convolutional layers with anchor boxes, Spatial Pyramid Pooling (SPP) block, Darknet-53 architecture, and an improved loss function [34]. Unlike previous versions, YOLOv4 was introduced by different authors. A. Bochkovskiy et al. enhanced YOLO's performance by utilizing CSPDarknet53 architecture, Bag-of-Freebies, Bag-of-Specials, mish activation function, Weighted-Residual-Connections (WRC), Spatial Pyramid Pooling (SPP), and Path Aggregation Network (PAN) [35].

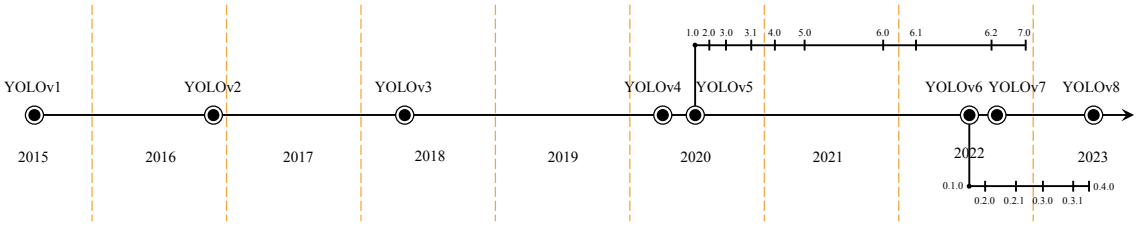


Figure 2.1: **YOLO release timeline.** YOLOv5 and YOLOv6 have ten and six released variants, respectively.

In 2020, Ultralytics introduced YOLOv5 in five different sizes ranging from nano to extra large [36]. YOLO underwent major modifications ranging from new backbone architecture to automated hyper-parameter optimization. In the backbone, YOLOv5 utilizes a new CSPDarknet53 structure [37] which is constructed based on Darknet53 with added Cross Stage Partial (CSP) strategy. The design of YOLOv5's Neck takes advantage of CSP-PAN [38] and a faster variation of SPP block (SPPF). The output is generated through the Head that uses the YOLOv3's Head structure. The structure of YOLOv5l is illustrated in Figure 2.2 in which CSPDarknet53 contains C3 blocks which are the CSP-fused modules. CSP strategy partitions the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy. Therefore, the C3 module can effectively handle redundant gradients while improving the efficiency of information transfer within residual and dense blocks. C3 is a simplified version of BottleNeckCSP and is currently used in the latest variant of YOLOv5. For comparison, the design of C3 and BottleNeckCSP blocks are depicted in Figure 2.3. Overall, these modifications have enabled YOLOv5 to achieve state-of-the-art performance on several benchmarks for object detection, including the COCO dataset. Also, different model sizes provide the user with the opportunity to choose based on their need.

In 2022, YOLOv6 was released by Meituan, featuring enhancements led by a Bi-directional Concatenation (BiC) module, an Anchor-Aided Training (AAT) strategy,

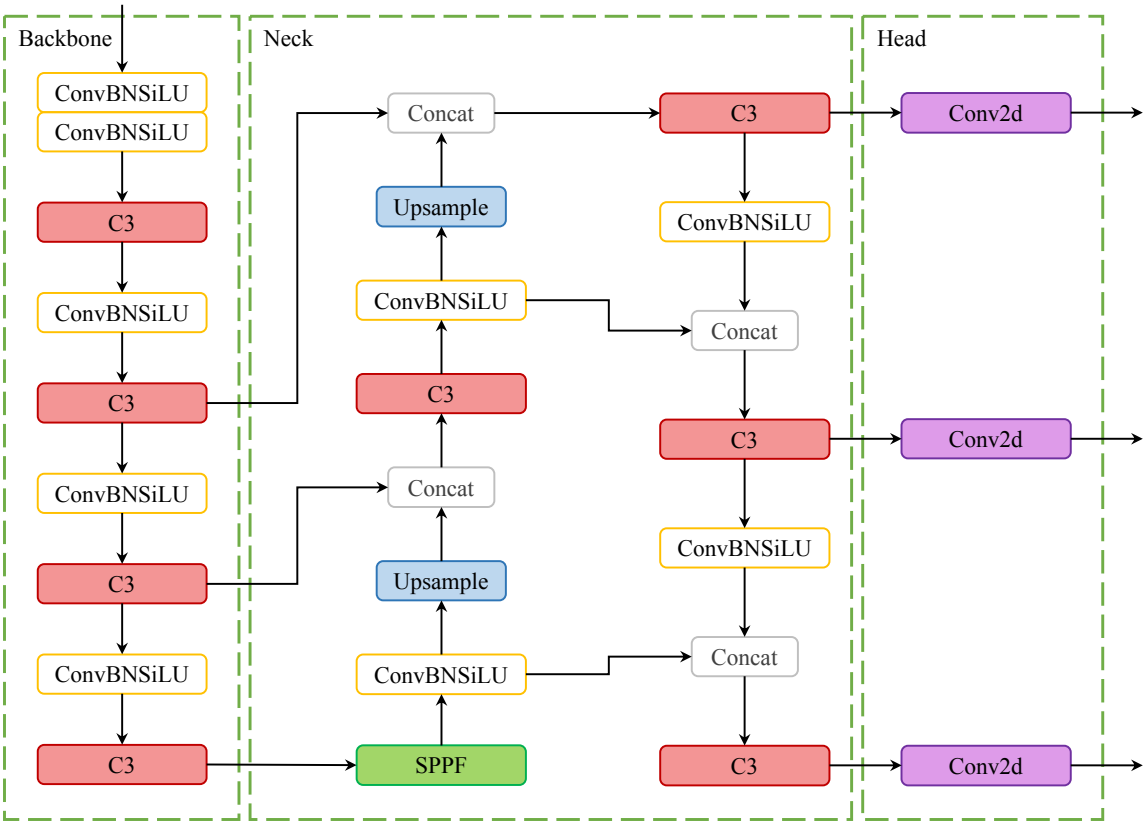
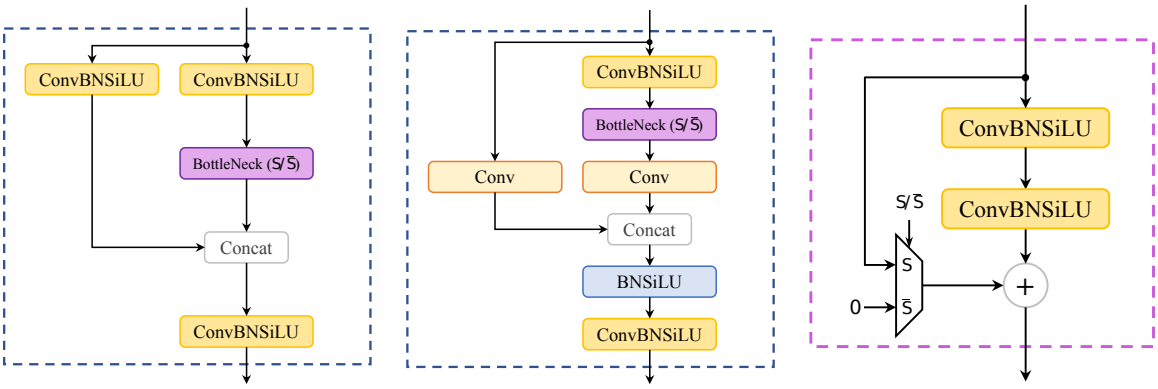


Figure 2.2: **YOLOv5l architecture.** SPPF represents a computation-efficient version of the Spatial Pyramid Pooling, which was originally implemented in YOLOv3; C3 uses the new CSP-combined module whose details are illustrated in Figure 2.3.

and a new backbone and neck design [39]. After a very short period, introduced by the original authors, YOLOv7 [40] was a breakthrough. Wang et al. proposed a Bag-of-Freebies, a compound model scaling method, and an extended ELAN architecture to expand, shuffle, and merge cardinality. The Bag-of-Freebies refers to a planned re-parameterized convolution inspired by ResConv[41], an extra auxiliary head in the middle layers of the network for Deep Supervision [42], and a soft label assigner guiding both the auxiliary head and the lead head by the lead head prediction. Lastly, Ultralytics presented YOLOv8 in 2023 with several alternations in the backbone, neck, and head [11]; A C2f module is used instead of C3; a decoupled head provides the output; and the model directly predicts the center of the object instead of the anchor



(a) C3 structure (b) BottleNeckCSP structure (c) BottleNeck structure

Figure 2.3: **Structure of C3 and BottleNeckCSP modules**. Using the CSP strategy enables the C3 module to strengthen information flow with residual and dense blocks while addressing redundant gradients. BottleNeck block, which is utilized in C3 and BottleNeckCSP and marked as purple, can have two configurations; $S/\bar{S}$. S denotes the active shortcut connection, while $\bar{S}$ characterizes a simple BottleNeck without any skip connection. C3 blocks of the Backbone use BottleNecks with shortcut connection, whereas those of the Neck do not.

box. The overall timeline of the YOLO releases is illustrated in Figure 2.1.

2.3 Neural Network Compression

The current trend in using and expanding over-parameterized models leads to higher accuracy; however, the required Floating-Point Operations (FLOPs) and parameters are dramatically increasing [43]. This issue impedes the deployment of complex models on edge devices owing to memory, power, and computational power limitations. To address this matter, one can employ cloud computing. However, running complex models on cloud services may not be a feasible option due to 1) the cost of the network: transferring image data to the cloud consumes relatively large network bandwidth; 2) Latency in time-critical tasks: access delay to cloud services is not guaranteed; 3) Accessibility: cloud services rely on the access of devices to wireless communications which can be disrupted in many environmental circumstances [44]. Hence, in many

cases, edge computing emerges as a more fruitful resolution. Accordingly, various methods have been introduced to compress neural networks with the purpose of making large models deployable on edge devices. Model compression methods can fall under three categories; pruning, quantization, and knowledge distillation. In pruning, redundant parameters of a model with less importance are removed to obtain a sparse/compact model structure. Quantization involves representing models' activations and weights using lower-precision data types. Finally, knowledge distillation refers to employing a large and accurate model as a teacher to train a small model using soft labels supplied by the teacher model(s) [45, 46].

We focus on pruning and quantization methods since they are widely used as modular compression techniques while utilizing knowledge distillation requires having two models or modifying the structure of the target network.

2.3.1 Pruning

Neural network pruning was initially proposed in Optimal Brain Damage [47] and Optimal Brain Surgeon [48]. They both rely on a second-order Taylor expansion to estimate parameter importance for pruning i.e. the Hessian matrix should be partially or completely computed in the mentioned methods. However, other criteria can be employed to identify the importance, also called *saliency*, of parameters. Hypothetically, while the best criterion would be an exact evaluation of each parameter's effect on the network, such evaluation is excessively costly to compute. Therefore, other evaluations, including ℓ_n-norm, the mean or standard deviation of feature map activation, batch normalization scaling factor, first-order derivative, and mutual information, can be utilized for saliency analysis. In the following section, we will discuss such saliency evaluation methods.

Saliency Criteria for Pruning

Saliency criteria refer to measures or metrics used to determine the importance or relevance of individual weights, neurons, filters, or a group of weights in a neural network based on certain characteristics or properties of the network. This review refers the reader to [49] for a detailed review of saliency criteria.

ℓ_n-norm. Pruning a model based on ℓ_n-norm is the predominantly-used method over the scope of this review. Since weight values generally form a normal distribution with zero mean, this is an intuitive approach to select less important individual or structure of weights. The challenge in using this criterion is to define a threshold with which pruning can be performed. Such a threshold can be set statically for the whole network or for each layer. Also, one can approach it as a dynamic parameter and define schedulers for this threshold. For instance, [50] proposes a method that treats threshold scheduling as an implicit optimization problem and provides threshold schedulers upon using Iterative Shrinkage-Thresholding Algorithm (ISTA).

ℓ_n-norm is usually combined with sparse training of the network to push parameters with the same effect to have similar values [49] (see Section 2.3.1). To do so, ℓ_1 or ℓ_2-regularization is usually added to the cost function, and parameters with a low ℓ_2 norm are pruned after (each step of) training [51, 52].

Feature map activation. When an activation function is used at the end of a layer, its output can be interpreted as the importance of parameters on the prediction. For instance, in the case of ReLU function, outputs closer to zero can be considered less salient and be chosen as candidates for pruning. Also, in a broader outlook, the mean or the standard deviation of a tensor of activation can indicate saliency [49, 53].

Batch normalization scaling factor (BNSF). Although it can be categorized as a fusion between ℓ_1-norm and feature map activation criteria, the BN scaling factor is used predominantly for pruning YOLOv5 and, more generally, for CNNs. Presented by [54], this method introduces a scaling factor γ for each channel and penalizes it during training to obtain a sparse network that can be pruned. The authors proposed the BN scaling factor as the γ needed for network compression. In their method, they penalize γ of the channels using ℓ_1-norm and then prune channels with near-zero scaling factors.

First-order derivative. Unlike the previous criterion, the first-order derivative metrics use the information provided during backpropagation via gradients. This class of criteria may combine information from activations to gradients as expressed in [49, 55].

Mutual information. The Mutual Information (MI) between the parameters of a layer and the prediction or that of another layer can represent saliency. In [56], authors tried to minimize the MI between two hidden layers while maximizing it between the last hidden layer and the prediction.

Granularity of Pruning

The granularity of pruning defines what kind of parameters of the model are to be pruned. Broadly, pruning can be done either in a structured or unstructured way.

Unstructured pruning. Unstructured or fine-grained pruning refers to when goal parameters for pruning are weights of the model without considering their location in the associated tensor or layer. In weight pruning, unnecessary weights are identified through saliency evaluation and masked out or removed afterward. Since eliminating

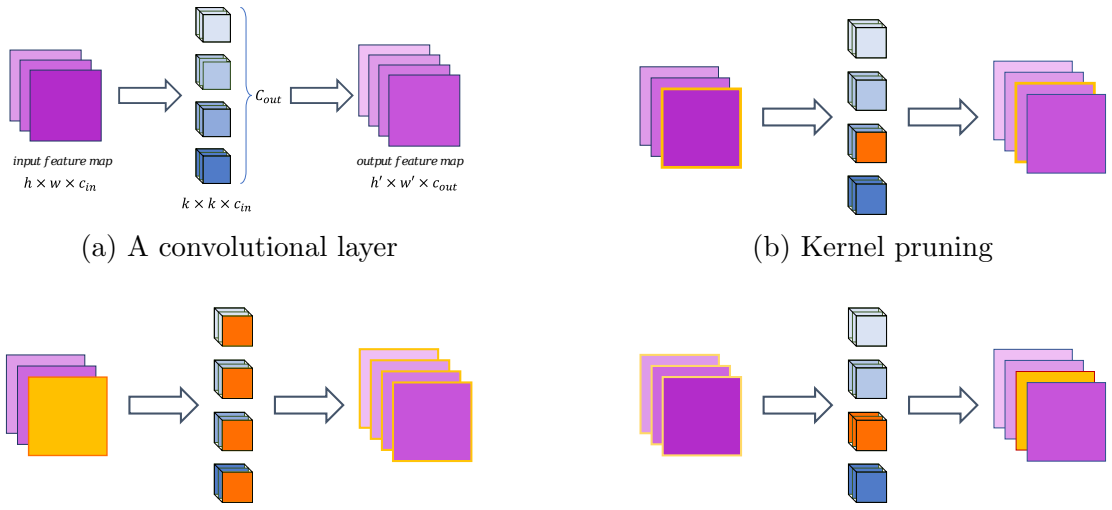
the weights may impair the model architecture, they are mostly masked out during this process [53]. While masking out the weights instead of removing them increases the memory usage during training, the information of the masked weights can be used at each step to compare the pruned model with the original one. fine-grained pruning is not always beneficial as it requires special hardware to take advantage of such irregular sparse operations [57]. While higher compression ratios could be achieved using unstructured pruning, storing the index of pruned weights may result in higher storage usage [58, 59].

Structured pruning. Unlike the previous category, model weights can be pruned based on their structure. Structured pruning observes patterns in the weights tensors when evaluating their importance so that they can be described with low indexing overhead, such as strides or blocks. In a convolutional layer, j-th channel is obtained through convolving the j-th filter with the input feature map. So groups of parameters such as filters, channels, or kernels can be selected for structured pruning. Figure 2.4 depicts the difference between these structural pruning paradigms.

Channel-based pruning. It aims at removing the weight filters leading to the j-th channel(s) of the output feature map in each layer. Many existing channel pruning techniques utilize the ℓ_2-norm as the criterion for identifying the least important tensor of weights. However, there is a dispute regarding the impact of this process on the overall model structure. In the [60], the authors stated that the process of channel pruning resulted in little damage to the model structure. Conversely, in the [61], it was observed that channel pruning led to a drastic change in the network structure. Nonetheless, it is possible to mitigate structural damage by masking the parameters rather than completely eliminating them. However, this approach may not result in any savings during training since the entire model needs to be stored in memory.

Filter-based pruning. Filter-based pruning eliminates the weights corresponding

to the i-th channel of the input feature map. That is, pruning a specific kernel, i-th, of all filters in a convolutional layer. The structural damage is minimal in this pruning method, and the model can be treated similarly to the original one since the number of output channels remains intact [43]. It is worth mentioning that: 1) channel-based pruning at l-th layer is equivalent to filter-based pruning at *(l+1)*th layer, and 2) Filter pruning is not equivalent to filter-based pruning. In filter pruning, one or some of the filters of a layer are pruned which can be categorized as channel-based pruning from the granularity perspective.



(a) A convolutional layer

(b) Kernel pruning

(c) Filter-based pruning

(d) Channel-based pruning

Figure 2.4: **Pruning granularity.** The dimension of a convolutional layer is shown in (a). Kernel, Filter, and Channel-based pruning are illustrated in (b), (c), and (d), respectively. The orange plane characterizes the pruned parameters, and the yellow bounding indicates the affected features.

Kernel-based pruning. In this category, all parameters over a kernel of a filter in l-th layer, which connects the i-channel of the input feature map and j-th channel of the output feature map, are pruned [62]. This pruning granularity would not impair the model structure. Interested readers may refer to [43] for a thorough survey of pruning granularity.

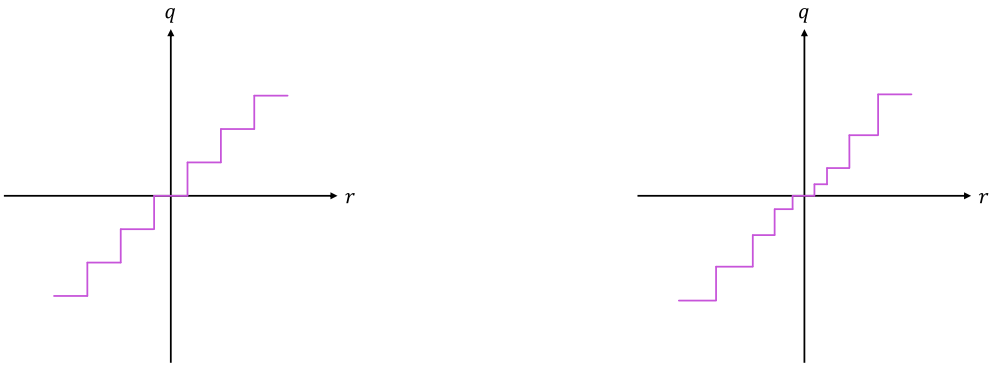
Regardless of the pruning granularity and the saliency criteria, the pruning pro-

cess can be performed either in a one-shot manner or iteratively. In one-shot pruning [63–65], less salient parameters are removed/masked prior to or after training. In post-training pruning, network performance may drop permanently, while iterative pruning, takes the performance drop into consideration and re-trains the network. Compared to one-shot pruning, although iterative pruning is more computation- and time-consuming, it can prevent the accuracy drop or even enhance it in some cases. Moreover, some methods modify the network cost function, such as adding regularization [54, 66], to make the model more suitable for pruning. Consequently, they cannot be used as post-training pruning.

2.3.2 Quantization

Neural network quantization aims to represent the weights and activations of a deep neural network with fewer bits than their original precision, which is usually a 32-bit single-precision floating-point (FP32). This process is done while the effect on the model performance and accuracy are kept minimal. Quantization, by exploiting faster hardware integer instructions, can reduce the size of the model and improve inference time [67]. In [46], Gholami et al. surveyed different aspects of neural network quantization, which includes the theoretical details of this topic. Here, the key points of quantization are introduced and discussed briefly.

Without loss of generality, the concept of quantization is explained on a real finite variable which can represent weights or activations in a neural network. Assuming $r \in \mathbb{R}$ is a finite variable limited to the range of $\mathbb{S}$, the goal is to map its values to q with a set of discrete numbers that lie in $\mathbb{D} \subset \mathbb{R}$. Before mapping, one may choose to clip the range of input r to a smaller set of $\mathbb{C} \subseteq \mathbb{S}$.



(a) uniform quantization (b) non-uniform quantization

Figure 2.5: **Quantization Interval.** (a) indicates the uniform quantization, while (b) depicts the non-uniform quantization in which neither the interval in the real number nor the quantization steps are evenly distributed.

Quantization Interval: Uniform and Non-uniform

Uniform quantization maps r into a set of evenly spaced discrete values, while in non-uniform quantization, the distance between discrete values is not necessarily equal. Through non-uniform quantization, one can better capture the vital information from the weights and activation distributions because, for instance, more closely-spaced steps can be allocated to denser areas of distribution. Consequently, although employing non-uniform quantization requires more design than the uniform approach, it may achieve a lower accuracy drop [68]. Also, since the distribution for weights and activations generally tends to be bell-shaped with long tails, non-uniform quantization can lead to better results [69–71]. Figure 2.5 demonstrates the disparity between the aforementioned quantization schemes.

Static and Dynamic Quantization

For a set of inputs, the clipping range $\mathbb{C} = [a, b]$, where $(a, b) \in \mathbb{R}$, can either be determined dynamically or statically. The former computes the clipping range dynamically for each input, whereas the latter uses a pre-calculated range to clip all the

inputs. Dynamic quantization reaches higher accuracy compared to static one, but the computational overhead is significantly high.

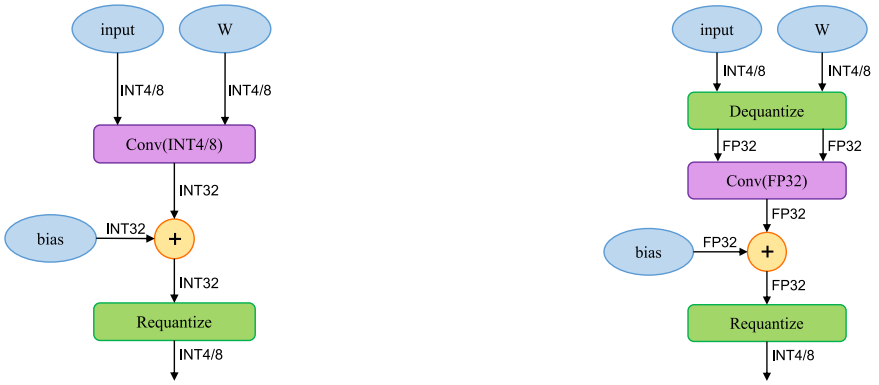
Quantization Scheme: QAT and PTQ

Quantizing a trained model can negatively impact the accuracy of the model due to cumulative numerical errors. To recover this drop in performance, network parameters often need adjustment. Therefore, quantization can be performed in two fashions; Quantization Aware Training (QAT), which refers to retraining the network, or Post Training Quantization (PTQ), which does not include re-training. In QAT, the forward and backward passes of the quantized model are performed in floating points, and network parameters are quantized after each gradient update. On the other hand, PTQ performs quantization and parameter adjustment without re-training the network. Compared to QAT, this method usually suffers from the model's accuracy degradation, but its computational overhead is considerably lower. Generally, PTQ uses a small set of calibration data to optimize the quantization parameters and then quantizes the model [72]. As PTQ relies on minimal information, it is often impossible to achieve lower than 4 or 8 bits precision while maintaining accuracy [73].

Quantization Deployment Scheme

Once the model is quantized, it can be deployed using *fake quantization* (also called simulated quantization) or *integer-only* quantization (also known as fixed-point quantization). In the former, weights and activations are stored in low precision, but all the operations ranging from addition to matrix multiplication are performed in floating point precision. While this approach requires constant dequantizing and quantization before and after floating point operations, it favors the model's accuracy. However, in the latter, operations, as well as weights/activation storing, are performed using

low-precision integer arithmetic. In such a fashion, the model can take advantage of fast integer arithmetic enabled by most hardware. Figure 2.6 illustrates the disparity between PTQ and QAT deployment for a single convolutional layer.



(a) Fixed-precision quantization (b) Simulated quantization

Figure 2.6: Inference comparison between fixed-precision and simulated quantization in a convolutional layer.

Overall, model compression methods have gained significant attention in recent years, and their applications are becoming more specific. In this section, a review of the commonly used pruning and quantization approaches applied to YOLOv5 is presented and then categorized from different aspects. This review identifies the gap in adapting pruning and quantization methods specifically for YOLOv5. This matter is of importance since the application of those compression methods to YOLOv5 requires further exploration and optimization due to many structural interconnections. This review serves as a resource for researchers interested in the practical deployment of model compression methods, specifically pruning and quantization, on YOLOv5 and its subsequent newer versions. With the advent of YOLOv8, YOLO has experienced a new borderline in terms of performance, shown in Figure 2.7, which pushes the boundary of object detection and establishes a new SOTA performance. However, the reduction in terms of size and FLOPs is relatively insignificant which necessitates the need for applying pruning and quantization on newer YOLOs when the limitation

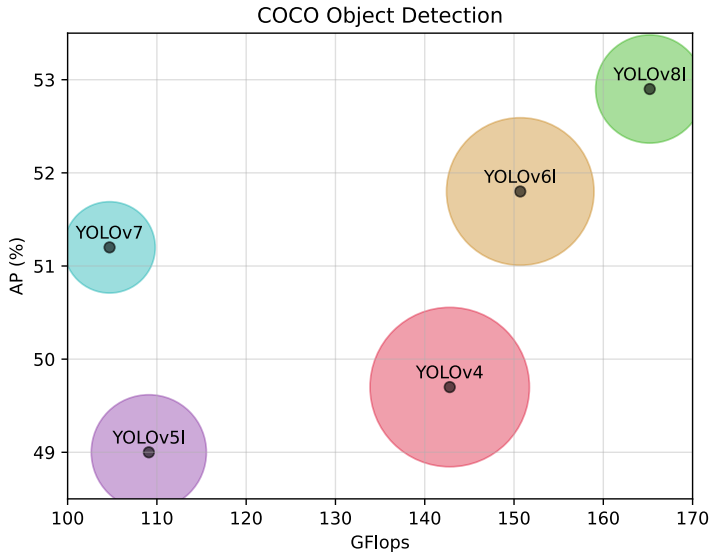


Figure 2.7: **Performance characteristic of four different YOLO versions.** The horizontal and vertical axes represent the number of GFlops and the average $AP^{0.5}$ on the COCO 2017 validation dataset with an input size of 640. The radius of circles denotes the relative size of the models.

in hardware is pivotal. Therefore, our review is extendable to subsequent versions of YOLO and can be used as a guideline for researchers interested in compressing any version of YOLO.

2.4 Motion Modeling

In multi-object tracking (MOT), accurately modeling the motion of objects is crucial for predicting their future positions, especially in scenarios involving camera motion and occlusions. Traditional SORT-based tracking approaches often use a constant velocity model to represent and predict target locations within the frame. While this simple model can accurately represent target movement in static-camera environments free from occlusion, its generalizability diminishes in scenarios involving moving cameras, particularly when occlusion arises.

Several researchers have attempted to address the limitations of the constant velocity model by incorporating more sophisticated motion modeling techniques. These techniques can broadly be classified into methods that use motion cues from objects and those that implicitly utilize ego-motion information from the camera or vehicle [10, 74–77]. One notable approach is the Motion-Aware Tracking (MAT) algorithm [10], which utilized the Enhanced Correlation Coefficient Maximization (ECC) [78] model to estimate the rotation and translational motion of the camera. It also employs Kalman Filter to predict the location of the targets. Another significant contribution is the Exploit Motion Model [74], which incorporates object motion information to enhance tracking performance. This method uses motion cues to improve the robustness of the tracker against occlusions and dynamic camera movements. The Multi-Cue Multi-Object Tracker (MC-MOT) [75] combines multiple motion cues, including object and camera motion, to provide a more reliable motion model. This approach leverages the interactions between different objects and the camera to improve tracking accuracy in complex environments. SiamMOT [76], a Siamese network-based tracker, integrates motion modeling directly into the tracking process. It uses a combination of appearance and motion features to maintain robust tracking performance, even in challenging scenarios with significant occlusions and dynamic camera movements. A motion-aware matching module was presented in [77] in order to match tracks with the new observations using motion feature. This approach helps in maintaining track continuity even when objects undergo complex motion patterns.

The integration of ego-motion data allows the Kalman Filter to account for the dynamic movements of the camera, providing a more accurate prediction of the target's future locations. This approach is particularly beneficial in autonomous driving applications, where the ego vehicle's motion significantly impacts the observed trajectories of surrounding objects.

In order to achieve such ego-motion information from solely mono camera stream, one can opt to utilize Visual Odometry (VO) or Simultaneous Localization And Mapping (SLAM). Depth estimation from mono camera also provide pseudo spatial understanding of the scene and object.

2.5 Visual Odometry and Depth Estimation

Integrating visual odometry and depth estimation into MOT systems enhances the accuracy and robustness of tracking by providing additional spatial and motion information. Visual odometry helps in accurately estimating the camera's motion, which is crucial for compensating for ego-motion in tracking algorithms. Depth estimation provides three-dimensional information about the scene, enabling more accurate object localization and trajectory prediction.

2.5.1 Visual Odometry

Visual Odometry (VO) involves estimating the trajectory of a moving camera by analyzing the visual data captured over time. It is a key technique in autonomous driving, robotics, and augmented reality applications. VO algorithms typically involve feature extraction, matching, and motion estimation. Features are extracted from consecutive frames and matched to compute the relative motion of the camera between frames. The matched features are then used to estimate the camera's trajectory using optimization techniques. Various methods have been proposed, leveraging different aspects of image data, such as optical flow, feature matching, and deep learning. One of the significant contributions to the field is TANDEM [79], which introduced a real-time monocular pose estimation and dense depth mapping framework that combines learning-based multi-view stereo (MVS) with traditional optimization-based

VO. The system uses a novel MVS network called Cascade View-Aggregation MVS-Net (CVA-MVSNet) to predict dense depth maps by constructing 3D cost volumes using adaptive view aggregation. TANDEM considers a sliding window of keyframes for pose estimation and makes photometric bundle adjustment. It also utilizes a global depth model created by interatively extracting dense depth maps and proposes dense direct image alignment with depth maps. Another noteworthy method is SfMLearner [80], developed by Zhou et al. This framework presents an unsupervised learning approach that estimates depth and ego-motion from monocular videos. SfM-Learner employs a convolutional neural network (CNN) to predict depth maps and relative camera poses, facilitating effective monocular VO. By using a combination of photometric consistency loss and a novel depth smoothness term, SfMLearner is able to learn depth and ego-motion without ground truth labels. This method has been pivotal in advancing unsupervised learning approaches for VO, demonstrating that reliable depth and motion estimation is possible using only monocular images. In 2021, Wang et al. presented TartanVO [81], a learning-based VO system trained on the TartanAir dataset. TartanVO generalizes well across different environments but faces challenges in scale recovery. The system uses a combination of depth and pose estimation networks to achieve robust performance in various settings. Following the previous work, DytanVO [82] was proposed, which uses motion segmentation in order to mask moving objects and output more accurate visual odometry estimation. This method takes advantage of an optical flow network along with a motion segmentation network based on the U-net architecture to construct additional channels for a CNN-based pose network. It uses a feedback system to iteratively output the up-to-scale relative pose vectors. DytanVO's ability to handle dynamic scenes makes it a valuable contribution to VO in challenging environments. Depth-VO-Feat [83], introduced by Zhan et al. in 2018, combines VO with monocular depth estimation

using deep feature reconstruction. This method offers an unsupervised framework that leverages CNN that takes in a temporal image pair and outputs the transformation vector. AirVO [84], presented by Gao et al. in 2023, emphasizes illumination robustness and accuracy in stereo visual odometry. It integrates point and line features for improved performance in varying lighting conditions, making it particularly useful for outdoor and dynamic environments. DF-VO [85, 86], developed by Zhan et al. in 2020, integrates depth and optical flow for visual odometry. It finds the 2D-2D correspondence through dense optical flow prediction and 3D-2D correspondence through monocular depth estimation. This approach enhances pose estimation accuracy through heuristic methods, offering a robust solution for dynamic scenes even when the camera motion is pure rotational. SimVODIS [87], introduced by Kim et al. in 2022, integrates pose estimation with depth estimation and instance segmentation through a multi-task learning framework providing comprehensive scene understanding. It relies on three consecutive images from a monocular camera and various neural network modules starting with a feature pyramid network to generate the camera pose, scene inverse depth map, and instance segmentation masks. Francani et al. introduced DPT-VO in 2022, which uses a Dense Drediction Transformer for scale estimation in monocular visual odometry. Its accurate depth estimation helps reducing the scale drift issue which ties closely with monocular visual odometry. The same authors also developed TSformer-VO [88], a transformer-based solution for visual odometry from monocular images. It leverages video understanding techniques based on spatio-temporal self-attention mechanisms for end-to-end camera motion estimation. Finally, Structure-PLP-SLAM [89], presented by Wang et al. in 2023, integrates points, lines, and planes for efficient sparse mapping and localization. It supports monocular, RGB-D, and stereo cameras, enhancing flexibility in SLAM applications.

2.5.2 Depth Estimation

Depth estimation involves determining the distance of objects from the camera using monocular or stereo vision techniques. Monocular depth estimation uses a single camera to predict depth information, while stereo vision uses two cameras to triangulate the distance of objects based on the disparity between the two images.

Monocular depth estimation has gained significant attention with the advent of deep learning techniques. One prominent method is the use of convolutional neural networks (CNNs) to estimate the depth based on a mono image. Methods like Monodepth [90] and its successors [91] have demonstrated the effectiveness of using CNNs for depth estimation, providing accurate depth maps without the need for stereo image pairs.

In 2019, struct2depth [92], was introduced by Casser et al. It provides an unsupervised learning framework for depth and ego-motion estimation from monocular video. It models the 3D scene and individual object motion, providing a more detailed understanding of dynamic scenes. The method uses a combination of photometric loss and geometric constraints to improve the accuracy of depth and motion estimates.

SC-Depth V1 [93, 94], presented by Bian et al. in 2021, outputs both depth and relative pose from monocular videos. It integrates these estimates into ORB2, a pseudo RGB-D SLAM system, to construct maps and optimize poses without absolute scale. The method uses a structure-from-motion approach to enhance depth and pose estimation. Building upon the same work, SC-DepthV3 was presented in 2023, focusing on scale-consistent depth estimation over video sequences. It combines temporal consistency with spatial depth estimation, providing more reliable depth maps across frames. The method uses a recurrent neural network to enforce temporal coherence, significantly improving the stability of depth predictions in video.

LKVOLearner [95], introduced by Wang et al. in 2018, proposes learning depth

from monocular videos using direct methods. Direct method in visual odometry refers to finding an optimum camera pose that minimizes the photometric error between the warped image and the reference image. LKVOLearner combines photometric errors and geometric constraints to train the network unsupervised, achieving accurate depth estimation. The method is particularly effective in handling diverse environments and lighting conditions.

Insta-DM [96], developed by Lee et al. in 2021, focuses on learning monocular depth in dynamic scenes while considering the depth consistency by gaining instance projection awareness. It segments dynamic objects and estimates their depth and pose separately, providing a more accurate depth map for both static and dynamic elements in the scene. The method uses a combination of instance segmentation, pose estimation, and depth estimation networks and trains them in a joint end-to-end framework without supervision. Robust-CVD [97], presented in 2021, introduces an algorithm for estimating consistent dense depth maps and camera poses from monocular videos. The method utilizes CNN to estimate depth per frame, then it optimizes the camera pose and depth deformation in a joint manner to finally construct an enhanced depth map. It addresses both the large-scale and fine-scale misalignment problem using this architecture. MiDas [98], introduced by Ranftl et al. in 2022, focuses on mixing different datasets to perform zero-shot transfer learning in order to achieve robust and accurate monocular depth estimation. It aims to generalize well across different datasets without fine-tuning, providing a versatile solution for depth estimation. Based on the previous work, ZoeDepth [99] was presented by Bhat et al. in 2023. It proposes zero-shot transfer learning by exploiting both relative and metric depth estimation. The method achieves high performance in depth estimation without requiring dataset-specific training, showcasing its potential for practical applications. DEEPV2D [100], presented by Teed and Deng in 2020, proposes an end-

to-end deep learning network for predicting depth from video stream. The method interleaves two stages: motion estimation and depth estimation, enhancing the overall prediction accuracy through a comprehensive approach. Consistent Video Depth Estimation [101], developed by Chen et al. in 2020, introduces an algorithm for reconstructing dense, geometrically consistent depth maps for all pixels in a monocular video. The method uses optical flow to find backward/forward correspondences in image pairs and employs them to extract geometric constraints in 3D which results in geometric consistency across frames.

One of the most promising approaches in relative and metric depth estimation is Depth Anything [102] developed by Facebook researchers in 2024. It introduces a practical solution for robust monocular depth estimation. The method enlarges the dataset by designing an engine to collect and automatically annotate large-scale unlabeled data, which results in an enhanced generalizability of the method. It employs data augmentation tools to create a challenging optimization target, pushing the model to learn extra visual knowledge. Depth Anything also ensure that the model benefits from rich semantic priors, which exists in the pre-trained encoders. The method demonstrates impressive generalization ability and sets new state-of-the-art results after fine-tuning with metric depth from NYUv2 and KITTI datasets.

Finally, MonoDepth2 [103] presents a novel approach to monocular depth estimation using a single image. This method builds upon previous unsupervised methods by incorporating a more robust training paradigm and loss functions, achieving state-of-the-art performance for its time. MonoDepth2 leverages an image pair for supervision during training, which allows it to learn depth even in the absence of ground truth data. The method introduces several key innovations that significantly improve depth estimation quality.

First, MonoDepth2 proposes a minimum reprojection loss with the core idea of

handling occlusions in depth estimation. Traditional self-supervised monocular methods often struggle with occlusions because the pixel correspondence might not exist in all views. The minimum reprojection loss addresses this by considering only the view where the pixel is visible, which leads to sharper and more accurate depth maps. Second, the method includes an auto-masking loss in order to invalidate the corrupted pixel which are determined by considering the effect of the camera motion assumptions. This auto-masking technique dynamically excludes pixels from the loss calculation if their appearance does not change significantly between frames, effectively dealing with stationary objects and regions where there is no relative motion between the camera and the environment. Third, MonoDepth2 reduces the visual artifacts in depth estimation by employing a full-resolution multi-scale sampling method. Instead of computing the photometric error on downsampled images, the depth predictions are upsampled to the input image resolution, computing all losses at the full resolution. This approach reduces artifacts such as texture copying and improves the overall accuracy of depth predictions. Lastly, MonoDepth2 uses a standard, fully convolutional U-Net architecture for depth prediction and a separate ResNet-based network for pose estimation. The system is trained using a combination of photometric reconstruction error and edge-aware smoothness loss. Experimental results on the KITTI dataset demonstrate that MonoDepth2 achieves state-of-the-art performance, outperforming previous self-supervised and some fully supervised methods.

2.6 State Estimation Filters

State estimation is a critical component in many dynamic systems, providing accurate and reliable information necessary for control, navigation, target tracking, and decision-making processes. While Kalman Filter is a popular choice for linear Gaus-

sian systems, several advanced filtering techniques have been developed to handle nonlinearities, non-Gaussian noise, and other circumstances. This section explores various advanced filtering techniques used for state prediction and estimation.

2.6.1 Kalman Filter

The Kalman Filter, introduced by Rudolf E. Kalman in 1960, is a recursive algorithm used for estimating the state of a linear dynamic system from a series of noisy measurements. It operates by predicting the future state of the system and updating this prediction based on imprecise and uncertain measurements. It estimates every state of the system, even the hidden ones, through this process which eventually minimizes the mean square error of the estimates. The mathematical formulation of the Kalman filter starts with the simple introduction of a dynamic system which is the main focus of this thesis.

There are a few variants of the Kalman filter with different properties. Here, the focus is essentially on the original Kalman filter and interested readers may refer to [104] for other variants such as Smoothed Estimates, Fixed-Lag Smoothing, and Fading Memory. Kalman Filter involves two main stages namely *prediction* and *update*. In the prediction stage, the state and covariance are projected forward in time while in the update stage, the predictions are corrected based on the new data measurements.

2.6.2 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a nonlinear version of the Kalman Filter. It linearizes the nonlinear system dynamics around the current state estimate using a first-order Taylor expansion. This linearization allows the EKF to handle a wider range of applications compared to the standard Kalman Filter.

2.6.3 Other Advanced Filters

In addition to the Kalman Filter and its extension to nonlinear systems through the EKF, several other advanced filters have been developed to address various complexities in state estimation. For instance, Particle Filters, or Sequential Monte Carlo methods, represent the state distribution using a set of particles. Each particle represents a possible state of the system, and weights are assigned based on the likelihood of the observed measurements [105]. The H∞ Filter is designed to provide robust estimation under worst-case scenarios. It minimizes the worst-case estimation error rather than the mean square error [106]. The Information Filter, also known as the inverse covariance filter, operates in the information form of the Kalman Filter. It uses the inverse of the covariance matrix and the information vector, which can be advantageous in certain computational scenarios [107]. The Moving Horizon Estimator is an optimization-based approach that solves a finite-horizon optimization problem at each time step to estimate the state, considering the most recent measurements and system dynamics [105].

Overall, each of these filtering techniques offers unique advantages and is suitable for different types of systems and applications. The selection of an appropriate one for state estimation depends on the specific application and its requirements. The Kalman Filter, with its simplicity and computational efficiency, remains a fundamental and widely-used method for linear systems with Gaussian noise. Its extension to nonlinear systems through the Extended Kalman Filter provides a practical approach for many real-world applications where the system dynamics are not perfectly linear. By choosing the Kalman Filter and its extended variant, this thesis leverages a well-established and efficient method for state estimation, ensuring robustness and accuracy in the target application of vision-based ego-motion aware MOT.

Chapter 3

Problem Definition and Formulation

3.1 Conformal Prediction

There are various methods in literature to quantify the uncertainty of an NN model; some of which such as Deep Evidential Regression [108] require model retraining while others like Conformal Prediction (CP) [109] do not.

Conformal Prediction is a general method to provide prediction intervals or prediction sets for any model. It is a powerful approach to quantify the model uncertainty by outputting a distribution-free coverage guarantee [110, 111]. The core idea is to fit a model in a training set and quantify the uncertainty in the future prediction by computing the scores on a hold-out validation dataset [109].

Given n training samples $\{(X_i, Y_i)\}_{i=1}^{n}$ which are drawn exchangeable from a joint distribution P_{XY}. A machine learning model, $\mathcal{M}$, is trained to predict the unknown value of Y_{n+1} based on a given test point X_{n+1}. The goal in conformal prediction is to construct a marginal distribution-free prediction interval $C(X_{n+1}) \subset \mathbb{R}$ that is

guaranteed to contain Y_{n+1} with a pre-specified miscoverage rate, α. More specifically:

$$\mathbb{P}\{Y_{n+1} \in C(X_{n+1})\} \geq 1 - \alpha \tag{3.1}$$

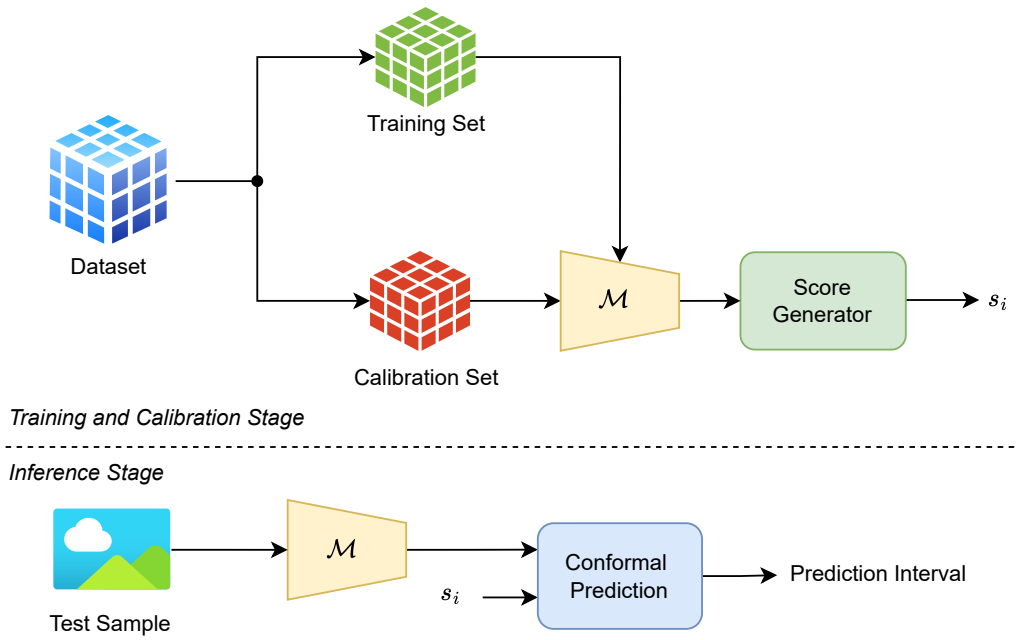


Figure 3.1: The overall framework of split conformal prediction

In the original/full conformal prediction, the model is invoked indefinitely many times while in the split/inductive variant, the dataset is split into two subsets. The latter requires residuals of the model inference on the validation set only [112–114]. Here the focus is exclusively on inductive conformal prediction which will be utilized in chapter 4. Depicted in Figure 3.1, the dataset is initially split into two subsets namely, a training set $\{(X_i, Y_i) : i \in \mathcal{S}_1\}$ and a calibration set $\{(X_i, Y_i) : i \in \mathcal{S}_2\}$. The regression model $\mathcal{M}$ is trained and fit to the training set such that:

$$\hat{\mu}(X) \leftarrow \mathcal{M}(\{(X_i, Y_i) : i \in \mathcal{S}_1\}) \tag{3.2}$$

Then, using a score function, the disparity between the prediction and actual output value is computed on the calibration set. This *nonconformity score* is larger when the agreement between these two values is worse:

$$s_i = s(Y_i, \hat{\mu}(X_i)), \quad i \in \mathcal{S}_2 \tag{3.3}$$

Considering the specified miscoverage rate α, the empirical $(1-\alpha)$-quantile of the calibration scores are acquired using:

$$Q_{1-\alpha} = (\frac{(1-\alpha)(1+|\mathcal{S}_2|)}{|\mathcal{S}_2|})\text{-th quantile of the } \{s_i; i \in \mathcal{S}_2\} \tag{3.4}$$

Finally, the prediction set for the new input is given by:

$$C(X_{n+1}) = \{y : s(y, \hat{\mu}(X_{n+1}) \leq Q_{1-\alpha})\} \tag{3.5}$$

which is guaranteed to satisfy equation 3.1.

It is worth noting that although this guarantee holds, the usefulness of the prediction set is mainly determined by the score function. A simple choice for regression problem is absolute residuals [115] such that :

$$r_i = |Y_i - \mu(X_i)| \quad i \in \mathcal{S}_2 \tag{3.6}$$

Consequently, the prediction interval after computing the quantile is given by:

$$C(X_{n+1}) = [\hat{\mu}(X_{n+1}) - Q_{1-\alpha}, \hat{\mu}(X_{n+1}) + Q_{1-\alpha}] \tag{3.7}$$

Although conformal prediction is known to provide prediction interval for distribution-free data. In cases where an assumption could be made on the data distribution,

conformal prediction can still be utilized to guarantee coverage within the miscoverage range. Assuming a normal distribution of the data, for instance, the standard deviation of the score is computed as follows:

$$\sigma = \sqrt{\frac{1}{|\mathcal{S}_2|} \sum_{i \in \mathcal{S}_2} (s_i - \bar{s})^2} \tag{3.8}$$

and the prediction interval that guarantees equation 3.1 is defines as:

$$C(X_{n+1}) = [\hat{\mu}(X_{n+1}) - z_{\alpha/2} \cdot \sigma, \hat{\mu}(X_{n+1}) + z_{\alpha/2} \cdot \sigma] \tag{3.9}$$

Conformal prediction will be employed on depth estimation in chapter 4 to quantify the uncertainty of the estimated depth from mono images.

3.2 Object Tracking

Multi-object tracking (MOT) is a critical task in computer vision, where the goal is to detect and track multiple objects across a sequence of video frames. This problem is fundamental to various applications, including autonomous driving, surveillance, sports analysis, and robotics. In this chapter, the MOT problem is defined and its underlying mathematical formulations is presented

The primary goal of MOT is to estimate the trajectories of multiple objects as they move through a scene. Given a sequence of video frames, MOT involves detecting objects in each frame and maintaining their identities across the frames, despite challenges such as occlusions, camera motion disturbances, changes in appearance, and interactions between objects.

Formally, let $\mathbf{I}_n$ denote the image frame at time n. The objective of MOT is to determine a set of trajectories $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$, where N is the number of

tracked objects, and each trajectory $\mathcal{T}_i$ consists of a sequence of object states $\mathbf{x}_t^i$ over time. The state $\mathbf{x}_t^i$ typically includes the position, velocity, and size of the object in the frame.

3.2.1 Object Detection

The first step in detection-based MOT is object detection, which involves identifying the objects present in each frame. Object detectors generate bounding boxes around detected objects and classify them into different categories. Let $\mathcal{D}_n = \{\mathbf{d}_n^1, \mathbf{d}_n^2, \ldots, \mathbf{d}_n^M\}$ denote the set of detections in frame $\mathbf{I}_n$, where M is the number of detections. Each detection $\mathbf{d}_n^j$ can be represented as:

$$\mathbf{d}_n^j = (\mathbf{b}_n^j, c_n^j, s_n^j)$$

where $\mathbf{b}_n^j$ is the bounding box, c_n^j is the class label, and s_n^j is the confidence score.

3.2.2 Data Association

Data association is the process of linking detections across frames to form object trajectories. This involves matching detections from the current frame t with the extrapolated location od tracked objects from the previous frames. The objective is to find the optimal assignment that maximizes the overall association score. Let $\mathcal{T}_{n-1} = \{T_{n-1}^1, T_{n-1}^2, \ldots, T_{n-1}^N\}$ denote the set of tracked object states at time $t-1$. The goal is to associate each detection $\mathbf{d}_{n|n-1}^j$ with one of the projected tracked objects $T_{n|n-1}^i$ or declare it as a new object. This can be formulated as an optimization problem:

$$\max_{\mathbf{A}} \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} \cdot S(T_{n|n-1}^i, \mathbf{d}_n^j)$$

subject to:

$$\sum_{j=1}^{M} a_{ij} \leq 1, \quad \sum_{i=1}^{N} a_{ij} \leq 1, \quad a_{ij} \in \{0, 1\}$$

Here, $\mathbf{A} = [a_{ij}]$ is the association matrix, and $S(T^i_{n|n-1}, \mathbf{d}^j_n)$ is the similarity score between the tracked object $T^i_{n|n-1}$ and the detection $\mathbf{d}^j_n$.

3.3 State Estimation and Prediction

Once the data association is performed, the next step is to estimate the current state of each tracked object and predict its future state. This is typically done using a filtering technique such as the Kalman Filter.

3.3.1 Kalman Filter-based MOT

Suppose that a dynamic system can be described by a linear, time-invariant state transition equation in discrete-time format:

$$\mathbf{x}_n = \mathbf{F}_n \mathbf{x}_{n-1} + \mathbf{G}_n \mathbf{u}_n + \mathbf{v}_n \tag{3.10}$$

where $\mathbf{x}_{n-1}$ denotes the state at time step $n-1$, $\mathbf{u}_n \in \mathbb{R}^p$ and $\mathbf{v}_n$ are the input vector and the process noise, $\mathbf{F}_n$ and $\mathbf{G}_n$ are the state and input transition matrices, and $\mathbf{x}_n \in \mathbb{R}^\eta$ indicates the system's state at the next time step. Assuming that measurements are fetched in the form of,

$$\mathbf{z}_n = \mathbf{H}_n \mathbf{x}_n + \mathbf{w}_n \tag{3.11}$$

$\mathbf{z}_n \in \mathbb{R}^q$ is the *observation* at time step n, $\mathbf{H}_n$ is the observation matrix, and $\mathbf{w}_n$ is the measurement noise. Kalman Filter operates under the assumption that all noises

are Gaussian, meaning they are uncorrelated and have zero mean:

$$\mathbb{E}[\mathbf{w}_n] = \mathbb{E}[\mathbf{v}_n] = 0 \tag{3.12}$$

The measurement and process noise covariance matrices are then positive-definite in the form:

$$\begin{cases} \mathbf{R}_n := \mathbb{E}[\mathbf{w}_n \mathbf{w}_n^T] \\ \mathbf{Q}_n := \mathbb{E}[\mathbf{v}_n \mathbf{v}_n^T] \end{cases} \tag{3.13}$$

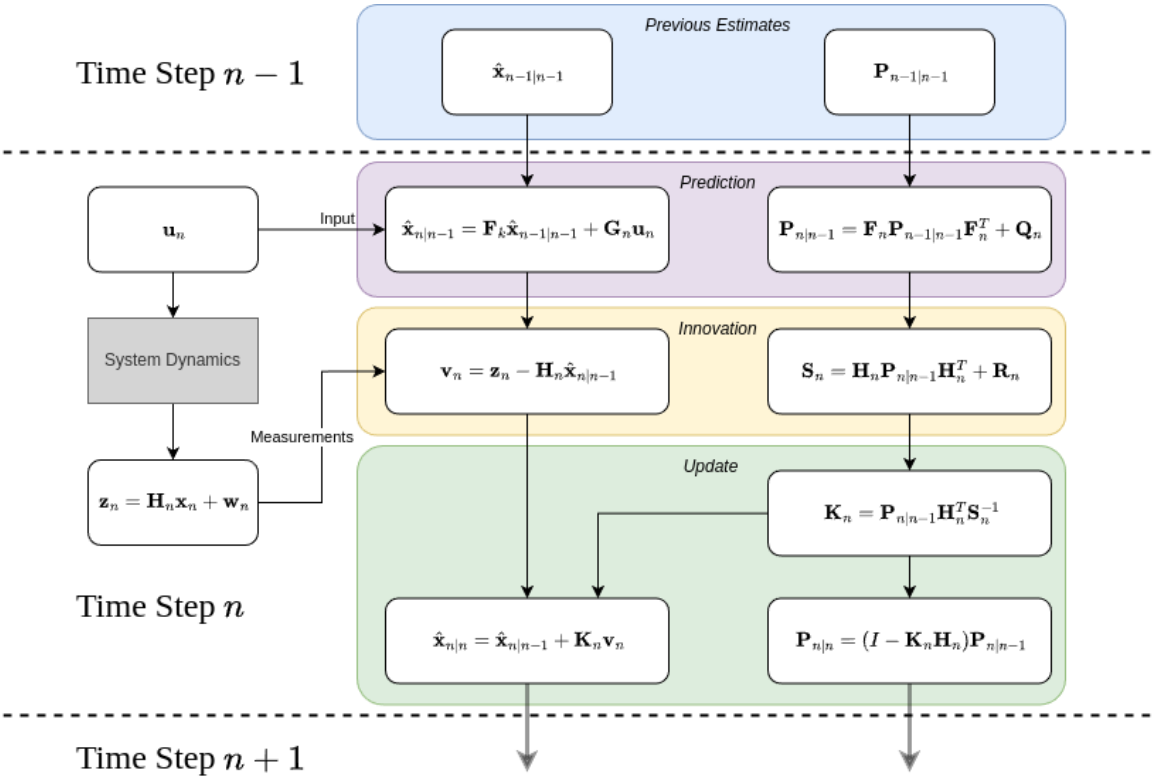


Figure 3.2: Block diagram of the Kalman Filter cycle

Consider the state estimation problem such that m observations $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_m$ and k inputs $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k$ are known where $k \geq m$. In the case where $k = m$, the primary goal in the original Kalman Filter is to find the state estimate, $\hat{\mathbf{x}}_{n|n}$ in terms of previous estimates $\hat{\mathbf{x}}_{n-1|n-1}$, and the latest data $\mathbf{u}_n$ and $\mathbf{z}_n$. This state estimator is optimized to minimize the mean-squared error $\mathbb{E}[||\hat{\mathbf{x}} - \mathbf{x}||^2]$ and the the covariance

$\mathbb{E}[(\hat{\mathbf{x}}-\mathbf{x})(\hat{\mathbf{x}}-\mathbf{x})^T]$. Kalman Filter's solution to this optimization problem is a recursive algorithm that is the best unbiased estimate of $\mathbf{x}_n$.

$$\begin{cases} \mathbb{E}[||\hat{\mathbf{x}} - \mathbf{x}||^2] \\ \mathbb{E}[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T] \end{cases} \tag{3.14}$$

The Kalman Filter consists of two primary stages: *prediction* and *update*. During the prediction stage, the state and covariance are projected forward in time. In the update stage, these predictions are adjusted based on new data measurements, refining the accuracy of the state estimation. The prediction and update stages are given as follows:

Prediction:

$$\hat{\mathbf{x}}_{n|n-1} = \mathbf{F}_k\hat{\mathbf{x}}_{n-1|n-1} + \mathbf{G}_n\mathbf{u}_n \tag{3.15}$$

$$\mathbf{P}_{n|n-1} = \mathbf{F}_n\mathbf{P}_{n-1|n-1}\mathbf{F}_n^T + \mathbf{Q}_n \tag{3.16}$$

Update:

$$\mathbf{K}_n = \mathbf{P}_{n|n-1}\mathbf{H}_n^T(\mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^T + \mathbf{R}_n)^{-1} \tag{3.17}$$

$$\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n(\mathbf{z}_n - \mathbf{H}_n\hat{\mathbf{x}}_{n|n-1}) \tag{3.18}$$

$$\mathbf{P}_{n|n} = (I - \mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1} \tag{3.19}$$

where $\mathbf{K}_n$ is the Kalman Gain at time step n.

As illustrated in Figure 3.2, the process begins with the initialization of the state estimates and the covariance. Then the filter extrapolates the system behavior at the next time step. Once the data measurement is received, the Kalman Filter computes the Kalman gain and corrects the predictions of the current state. The recursive

nature of the Kalman Filter allows it to process incoming data efficiently, making it suitable for embedded systems and real-time applications.

In KF-based MOT methods, a constant-velocity motion model is assumed for the objects in the frame. The object location in the image frame is iteratively predicted at each time step using the *predict* stage of Kalman Filter.

Afterward, newly detected objects are matched with the predicted positions of existing objects, resulting in new observations. In the next stage of the Kalman Filter known as the *update* stage, the Kalman gain, state estimate, and estimate covariance are all recalibrated

Implementation Consideration. The performance of the Kalman Filter relies heavily on the design of the measurement and process noise matrices. Poor value assignment in the component of those matrices will saturate the Kalman gain, pushing it close to 1 or 0. In this circumstance, the Kalman Filter barely compensates, if not at all, for modeling or measurements, respectively. One exciting feature of this filter, however, is that its performance for predicting or estimating the state is not affected by the scale of $\mathbf{R}$ and $\mathbf{Q}$. More specifically, if both the measurement noise covariance matrix, $\mathbf{R}$ and the process noise covariance matrix $\mathbf{Q}$ are multiplied by the same constant value, the relative balance between trusting the measurements and trusting the model predictions remains unchanged since the Kalman gain remain intact.

Many Kalman-filter-based approaches are built upon the foundation of SORT [5] with the following state definition:

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^\top \tag{3.20}$$

In this state definition, u and v denote the horizontal and vertical pixel locations of the center of the target object, while s and r denote the area and ratio of the

bounding box, respectively.

It is essential to recognize that this definition relies on the assumption of a constant velocity model for the tracked objects. Although this assumption may not accurately capture the behavior of all objects, it generally serves as a reasonable approximation in scenarios where the camera remains stationary, as evidenced by the MOT20 dataset. However, when the camera is mounted on a moving vehicle, the constant velocity assumption is frequently violated due to the motion of the ego vehicle. In particular, as the camera's movement becomes more dynamic, the accuracy of the predicted $\dot{u}$ and $\dot{v}$ values diminishes, even if the tracked object continues to move at a constant velocity in the real world.

3.3.2 Extended Kalman Filter

In cases a nonlinear dynamics are extracted for the target movements an Extended Kalman Filter (EKF) can be utilized. Using a first-order Taylor expansion, it linearizes the nonlinear system dynamics around the present state estimate. Let us consider the nonlinear system of the form:

$$\mathbf{x}_n = f(\mathbf{x}_{n-1}, \mathbf{u}_n) + \mathbf{v}_n \tag{3.21}$$

$$\mathbf{z}_n = h(\mathbf{x}_n) + \mathbf{w}_n \tag{3.22}$$

where f and h are nonlinear function. The EKF approximates these nonlinear functions using their first-order Taylor series expansions around the current estimate. The steps involved in the EKF are as follows:

Prediction stage

$$\hat{\mathbf{x}}_{n|n-1} = f(\hat{\mathbf{x}}_{n-1|n-1}, \mathbf{u}_n) \tag{3.23}$$

$$\mathbf{P}_{n|n-1} = \mathbf{F}_n \mathbf{P}_{n-1|n-1} \mathbf{F}_n^T + \mathbf{Q}_n \tag{3.24}$$

where $\mathbf{F}_n$ is the Jacobian of f with respect to $\mathbf{x}$, evaluated at $\hat{\mathbf{x}}_{n-1|n-1}$:

$$\mathbf{F}_n = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_{n-1|n-1}, \mathbf{u}=\mathbf{u}_n} \tag{3.25}$$

Update stage

$$\mathbf{K}_n = \mathbf{P}_{n|n-1} \mathbf{H}_n^T (\mathbf{H}_n \mathbf{P}_{n|n-1} \mathbf{H}_n^T + \mathbf{R}_n)^{-1} \tag{3.26}$$

$$\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n (\mathbf{z}_n - h(\hat{\mathbf{x}}_{n|n-1})) \tag{3.27}$$

$$\mathbf{P}_{n|n} = (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n) \mathbf{P}_{n|n-1} \tag{3.28}$$

where $\mathbf{H}_n$ is the Jacobian of h with respect to $\mathbf{x}$, evaluated at $\hat{\mathbf{x}}_{n|n-1}$:

$$\mathbf{H}_n = \left. \frac{\partial h}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_{n|n-1}} \tag{3.29}$$

Although this formulation may seem overly simple, EKF would perform badly if the system or sensor were modeled poorly, which might also cause EKF to diverge from the true states [104]. Consequently, utilizing EKF requires considerable engineering insight, especially in highly nonlinear systems.

3.4 Handling Occlusions and Birth/Death of Objects

In real-world scenarios, objects may become occluded, enter, or exit the field of view. The MOT system needs to handle these events appropriately.

Occlusion Handling: When an object is occluded, its state can be predicted using the motion model until it reappears. The tracking algorithm maintains a memory of the occluded objects and updates their states when they become visible again.

Birth and Death of Objects: New objects entering the scene are initialized as new tracks, while objects that have not been detected for a certain number of frames are terminated. This can be managed using a threshold on the number of missed detections.

3.5 Evaluation Metrics

The performance of MOT algorithms is evaluated using various metrics, including:

Multi-Object Tracking Accuracy (MOTA): Combines false positives, false negatives, and identity switches into a single measure.

$$\text{MOTA} = 1 - \frac{\sum_n (FN_n + FP_n + IDSW_n)}{\sum_t GT_n}$$

where FN_n is the number of false negatives, FP_n is the number of false positives, $IDSW_n$ is the number of identity switches, and GT_n is the number of ground truth objects at time step n.

Multi-Object Tracking Precision (MOTP): Measures the average overlap between the predicted and ground truth bounding boxes.

$$\text{MOTP} = \frac{\sum_n \sum_i d_n^i}{\sum_n c_t}$$

where d_n^i is the distance between the predicted and ground truth bounding boxes for object i at time step n, and c_n is the number of matches at time step n.

Identity F1 Score (IDF1): Measures the accuracy of the predicted identities.

$$\text{IDF1} = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFN + IDFP}$$

where $IDTP$ is the number of true positive identities, $IDFN$ is the number of false negative identities, and $IDFP$ is the number of false positive identities.

3.6 Summary

Multi-object tracking is a complex and challenging problem in computer vision that involves detecting, associating, and tracking multiple objects across video frames. By leveraging techniques such as object detection, data association, and state estimation, MOT systems can achieve robust performance in various applications. The mathematical formulations presented in this chapter provide a foundation for understanding and developing MOT algorithms.

Chapter 4

Methodology

In this chapter, the newly developed method is described in detail which serves as a prediction module in multi-object tracking. It begins with the underlying mathematical formulation to decouple the camera motion effect on the target location. Then, the ego-motion formulation is integrated into the Extended Kalman Filter followed by choice of pose and depth estimator. It finally explains how conformal prediction is utilized to provide EKF with real-time uncertainty awareness of the extracted depth.

4.1 Camera Motion

Consider a camera with a focal length f that views point P with real-world coordinates (x_P, y_P, z_P) and projects it onto the image plane at pixel coordinates (u, v) relative to the center of the image. Assuming we have a rectified image and a small forward or backward displacement of the camera along its principal axis, as illustrated in Figure 4.1b, the new pixel location of point P is:

$$u_2 = f \tan \gamma_2 \tag{4.1}$$

where tangent of γ_2 is extracted using tangent law in $\triangle PO_1O_2$:

$$\frac{d_1 - D}{d_1 + D} = \frac{\tan\frac{1}{2}\left(\pi - \gamma_1 - (\gamma_2 - \gamma_1)\right)}{\tan\frac{1}{2}\left(\pi - \gamma_1 + (\gamma_2 - \gamma_1)\right)} = \frac{\cot\frac{1}{2}\left(2\gamma_2 - \gamma_1\right)}{\cot\frac{1}{2}\left(\gamma_1\right)} \tag{4.2}$$

$$\gamma_2 = \frac{\gamma_1}{2} + \tan^{-1}\left(\frac{d_1 + D}{d_1 - D}\tan\left(\frac{\gamma_1}{2}\right)\right) \tag{4.3}$$

Following the goal of finding $\tan\gamma_2$, we take tangent of both sides:

$$\tan\gamma_2 = \frac{\tan\frac{\gamma_1}{2} + \left(\frac{d_1+D}{d_1-D}\ \tan\frac{\gamma_1}{2}\right)}{1 - \frac{d_1+D}{d_1-D}\ \tan^2\frac{\gamma_1}{2}} = \frac{2d_1\tan\frac{\gamma_1}{2}}{d_1 - D - (d_1+D)\tan^2\frac{\gamma_1}{2}} \tag{4.4}$$

$$\tan\gamma_2 = \frac{2d_1\tan\frac{\gamma_1}{2}}{d_1(1-\tan^2\frac{\gamma_1}{2}) - D(1+\tan^2\frac{\gamma_1}{2})} = \frac{2d_1\sin\frac{\gamma_1}{2}\cos\frac{\gamma_1}{2}}{d_1\cos\gamma_1 - D} \tag{4.5}$$

$$\tan\gamma_2 = \frac{d_1\sin\gamma_1}{d_1\cos\gamma_1 - D} \tag{4.6}$$

Finally, using Eq 4.1 and the definition of $\tan\gamma_1$, the new pixel location is:

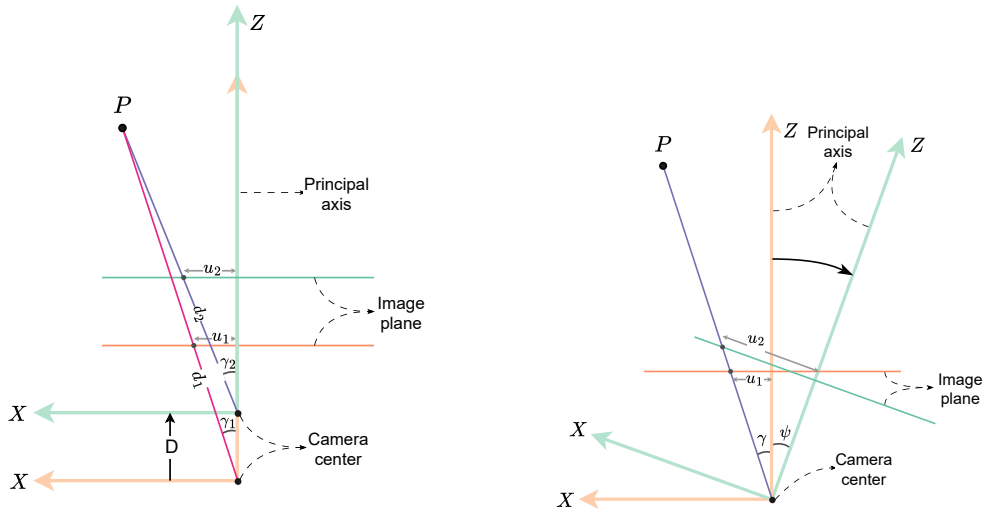
$$u_2 = f\frac{d_1\sin(\tan^{-1}(\frac{u}{f}))}{d_1\cos(\tan^{-1}(\frac{u}{f})) - D} \tag{4.7}$$

Similarly, the new vertical pixel location after such camera displacement is obtained from:

$$v_2 = f\frac{d_1\sin(\tan^{-1}(\frac{v}{f}))}{d_1\cos(\tan^{-1}(\frac{v}{f})) - D} \tag{4.8}$$

Moreover, the y-axis rotation of camera, as depicted in Figure 4.1a, also transfers the pixel location of P. This angular rotation only impacts the horizontal pixel location and is achieved through finding the $\tan\gamma_2$, which is also $\tan(\psi + \gamma_1)$:

$$\tan\gamma_2 = \frac{\tan\psi + \tan\gamma_1}{1 - \tan\psi\tan\gamma_1} \tag{4.9}$$



(a) (b)

Figure 4.1: The effect of camera rotation and translational displacement on pixel location of the point P.

Hence:

$$u_2 = f\frac{\tan(\psi) + \frac{u}{f}}{1 - \frac{u}{f}\tan(\psi)} \tag{4.10}$$

Now, let $\dot{D}$ represent the ego vehicle's forward translational velocity towards the normal vector of the image plane; u and v denote the horizontal and vertical pixel locations of the target relative to the center of the image, and $\hat{u}$ and $\hat{v}$ be their respective predicted values. According to Equations (4.7) and (4.8), the translational movement of the ego-vehicle influences the target object's location within the image frame as follows:

$$\hat{u}_{trans} = f\frac{d\sin(\tan^{-1}(\frac{u}{f}))}{d\cos(\tan^{-1}(\frac{u}{f})) - \Delta t\dot{D}} \tag{4.11}$$

$$\hat{v}_{trans} = f\frac{d\sin(\tan^{-1}(\frac{v}{f}))}{d\cos(\tan^{-1}(\frac{v}{f})) - \Delta t\dot{D}} \tag{4.12}$$

Also using Eq 4.10, the effect of ego-vehicle rotation on the pixel location of an object is:

$$\hat{u}_{rot} = f\frac{\tan(\Delta t\dot{\psi}) + \frac{u}{f}}{1 - \frac{u}{f}\tan(\Delta t\dot{\psi})} \tag{4.13}$$

where $\dot{\psi}$ is the ego-vehicle yaw rate.

Assuming a small time window between each two consecutive frames leads to linearized formulations of the form:

$$\hat{u}_{trans} \approx \frac{u\sqrt{u^2 + f^2}}{f} \cdot \frac{\dot{D}}{d}\Delta t + u \tag{4.14}$$

$$\hat{v}_{trans} \approx \frac{v\sqrt{v^2 + f^2}}{f} \cdot \frac{\dot{D}}{d}\Delta t + v \tag{4.15}$$

$$\hat{u}_{rot} \approx f\left(1 + \frac{u^2}{f^2}\right) \cdot \dot{\psi}\Delta t + u \tag{4.16}$$

It is worth noting that, in equations (4.14) and (4.15) the prediction of pixel location depends on the $\dot{D}/d$; meaning that our implementation can be completely agnostic of the actual metric scale as long as d and $\dot{D}$ belong to a same scaling space. Thus, we can rely on an up-to-scale depth and pose estimator that takes a mono camera image stream as the input. We do not consider using stereo cameras, as that approach is well-documented in the literature.

4.2 Camera Motion Integration with Kalman Filter

Considering the availability of scene depth map and camera motion information, we can now reformulate the Kalman Filter state definition to integrate this information into a base model. Following the similar format used by SORT in equation (3.20),

this thesis approximates the object motion by a constant velocity model; however, it decouples the effect of camera motion in this motion model. Each target object is then modeled using the following state definition:

$$\mathbf{x} = [u^l, v^t, u^r, v^b, \dot{x}^l, \dot{y}^t, \dot{x}^r, \dot{y}^b]^\top \tag{4.17}$$

Where (u^l, v^t) indicates the top-left corner of a bounding box, while (u^r, v^b) represents the bottom-right corner of that. Unlike SORT which uses derivatives of (u, v) as noted in 3.20, EMAP uses $\dot{x}^l, \dot{y}^t, \dot{x}^r$, and $\dot{y}^b$. These variables denote the projected horizontal and vertical pixel velocities of the corners of the target object bounding box caused solely by the object's motion in the world frame. Assuming that the movement of the ego-vehicle can be approximated by two independent parameters, i.e., orientation and forward displacement, state equation of the Kalman filter are as follows:

$$\hat{\mathbf{x}}_{\mathbf{n+1}} = \begin{bmatrix} I_{4\times4} & dtI_{4\times4} \\ 0_{4\times4} & I_{4\times4} \end{bmatrix} \hat{\mathbf{x}}_{\mathbf{n}} + \begin{bmatrix} G_n^\psi & G_n^D \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{D} \end{bmatrix} dt \tag{4.18}$$

where,

$$G_n^\psi = \begin{bmatrix} f\left(1 + \left(\frac{u_n^l}{f}\right)^2\right) \\ 0 \\ f\left(1 + \left(\frac{u_n^r}{f}\right)^2\right) \\ 0 \\ 0_{4\times1} \end{bmatrix}, \quad G_n^D = \begin{bmatrix} \frac{u_n^l\sqrt{(u_n^l)^2+f^2}}{fd_n} \\ \frac{v_n^t\sqrt{(v_n^t)^2+f^2}}{fd_n} \\ \frac{u_n^l\sqrt{(u_n^l)^2+f^2}}{fd_n} \\ \frac{v_n^b\sqrt{(v_n^b)^2+f^2}}{fd_n} \\ 0_{4\times1} \end{bmatrix} \tag{4.19}$$

When an associated detection of an object exists, Kalman Filter *update* phase is used to re-calibrate the state variables. Then, the *prediction* is performed to extrapolate the location of the target in the next frames. This process is illustrated in Figure 4.3

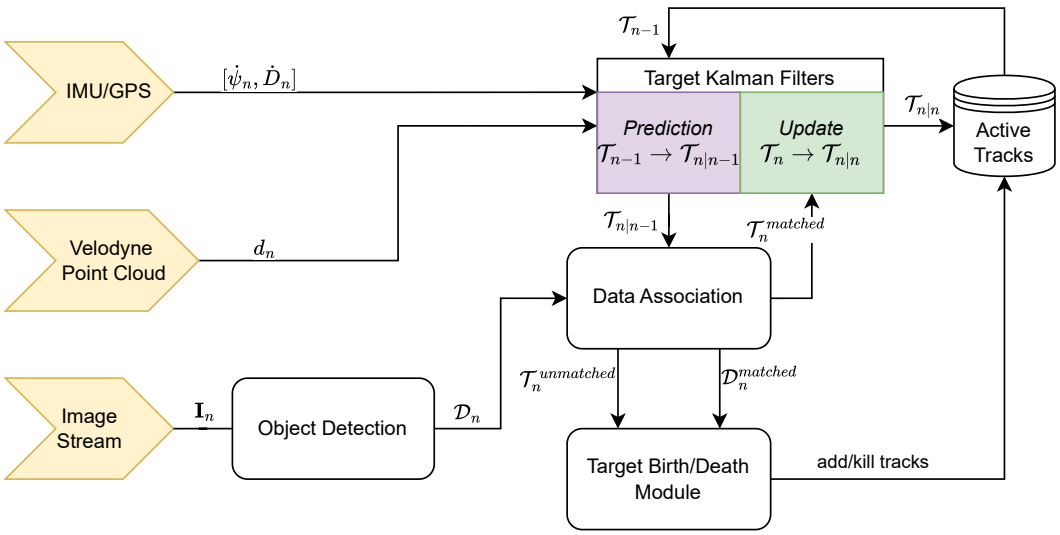


Figure 4.2: Overall Framework a Detection-based MOT with EMAP enabled

4.2.1 Pose and Depth Estimator

For depth and camera motion estimation, Monodepth2 [103] is modified and utilized, which trains a network via a self-supervisory signal derived from image reprojection errors. The method involves reconstructing an image, I_n, from the appearance of a prior image, I_{n-1}, using an encoder-decoder architecture to estimate camera motion between frames in an axis-angle format. This task is inherently challenging because many incorrect depth estimations per pixel could potentially recreate the desired view when combined with the correct relative pose.

Monodepth2 uses a U-net architecture, which features skip connections that help capture both high-level abstract features and detailed local information, to estimate

depth from a single monocular image. It converts the sigmoid output, σ, into depth using the formula $D = 1/(a\sigma + b)$, where a and b are parameters set to ensure D ranges between 0.1 and 100 units. Additionally, Monodepth2 includes a pose estimation module that decodes ego-motion from consecutive single-camera images. This module adjusts its inputs to accept pairs of images (six channels in total) and outputs a 6-DoF relative pose, scaled by 0.01 as per the authors' design. Although it is hypothesized to yield globally consistent depths, such consistency is not guaranteed by the architecture alone. This was tested by benchmarking Monodepth2 using a singular scale factor across all depth predictions, yielding promising results with minimal error ($< 10\%$ Rel error) compared to multi-scale assessments. In conclusion, I opt to modify and utilize Monodepth2 in our approach for tasks of pose estimation relative depth extraction. Figure 4.3 shows an overview of the Vision-Based EMAP when integrated into a KF-based MOT solution.

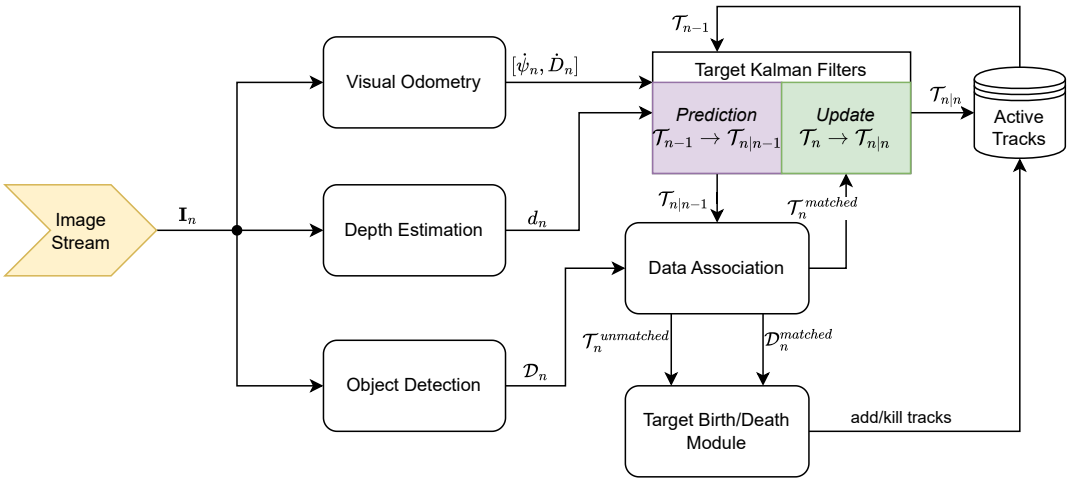


Figure 4.3: Overview of a VEMAP-enabled multi-object tracking

4.2.2 Conformal Prediction

Generally, conformal prediction involves calculating a nonconformity score for each instance in a data set, which measures how different or "non-conforming" each instance is compared to others. The scores are then used to determine prediction intervals that are likely to cover new observations with a specified probability. Following the basic explanation of conformal prediction in section 3.1, here adopt this approach to quantify the underlying uncertainty in mono camera depth estimation. In the problem of estimating depth from RGB mono camera, the samples are in the form of $\{(X_i, Y_i)_{i=1}^n\}$ where feature vectors $X \in \mathbb{R}^{3\times m\times n}$ and responses $Y \in \mathbb{R}^{m\times n}$. In order to achieve a prediction interval, the depth absolute residual on each pixel location, $(i, j) \in \mathbb{R}^{m*n}$, are computed on the validation subset of *KITTI Eigen* [116]:

$$r_{ij}^{(k)} = |Y_{ij}^{(k)} - \hat{Y}_{ij}^{(k)}|, \quad k \in \mathcal{S}_v \tag{4.20}$$

which is used to construct $\mathcal{R}_{ij} = \{r_{ij}^{(k)} : k \in \mathcal{S}_v\}$. Then, considering a desired miscoverage rate, α, the prediction is obtained by computing the standard deviation of the residuals. That is, using:

$$\sigma_{ij} = \sqrt{\frac{1}{|\mathcal{S}_v|}\sum_{k\in\mathcal{S}_v}(r_{ij}^{(k)} - \bar{r}_{ij})^2} \tag{4.21}$$

the prediction interval is then:

$$C(X_{n+1}, i, j) = [\hat{Y}_{ij} - z_{\alpha/2}\cdot\sigma_{ij}, \hat{Y}_{ij} + z_{\alpha/2}\cdot\sigma_{ij}] \tag{4.22}$$

where $z_{\alpha/2}$ is the standard score corresponding to the $(1-\alpha)$ confidence level.

Once the prediction interval is fetched for each pixel location of the image, the median of the quantiles within the object's bounding box is used to represent the

uncertainty of the object's depth

$$d_{uncert} = \text{median}(\{z_{\alpha/2} \cdot \sigma_{ij} : (i,j) \in \mathbf{b}_n^{\ell}\}) \tag{4.23}$$

4.3 Augmented-state Extended Kalman Filter

Expanding upon the previous Kalman filter formulation, the state space is augmented to contain object depth. This addition is justified because the depth is a distinct property of each object and is largely independent of other objects. Conversely, incorporating the ego yaw angle into the system states is impractical, as it represents shared information among all objects.

Similar to the other states in the system, I assume a constant velocity model for object depth and introduce the UVEMAP augmented state definition as follows:

$$\hat{\mathbf{x}}_{\mathbf{n+1}} = \begin{bmatrix} I_{5\times5} & dtI_{5\times5} \\ 0_{5\times5} & I_{5\times5} \end{bmatrix} \hat{\mathbf{x}}_{\mathbf{n}} + \begin{bmatrix} G_n^{\psi} & G_n^{D} \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{D} \end{bmatrix} dt \tag{4.24}$$

where,

$$G_n^{\psi} = \begin{bmatrix} f\left(1+\left(\frac{u_n^l}{f}\right)^2\right) \\ 0 \\ f\left(1+\left(\frac{u_n^r}{f}\right)^2\right) \\ 0 \\ 0 \\ 0_{5\times 1} \end{bmatrix}, \quad G_n^{D} = \begin{bmatrix} \frac{u_n^l\sqrt{(u_n^l)^2+f^2}}{fd_n} \\ \frac{v_n^t\sqrt{(v_n^t)^2+f^2}}{fd_n} \\ \frac{u_n^r\sqrt{(u_n^r)^2+f^2}}{fd_n} \\ \frac{v_n^b\sqrt{(v_n^b)^2+f^2}}{fd_n} \\ -\sqrt{\frac{f^2}{((u_n^l+u_n^r)/2)^2+f^2}} \\ 0_{5\times 1} \end{bmatrix} \tag{4.25}$$

This new definition also enables us to take advantage of real-time depth measurement uncertainty, if available, which is further explained in the next section. Using the augmented-state EKF in UVEMAP, the general process of UVEMAP is depicted in Figure 4.4

4.3.1 Uncertainty-aware EKF

The extracted uncertainty for depth is integrated into our EKF as follows:

$$\mathbf{R}_n = \text{diag}([(w_d\sigma_u)^2, (w_d\sigma_v)^2, (w_d\sigma_u)^2, (w_d\sigma_v)^2, d_{uncert}^2(n)]) \tag{4.26}$$

where w_d is the mean of the d_{uncert} over the whole calibration set. It is included in the measurement noise matrix in order to scale the noise and ensure the Kalman gain does not saturate. The Kalman filter is initialized with high uncertainty covariance for each object so that the filter applies more gain to settle around the measurement

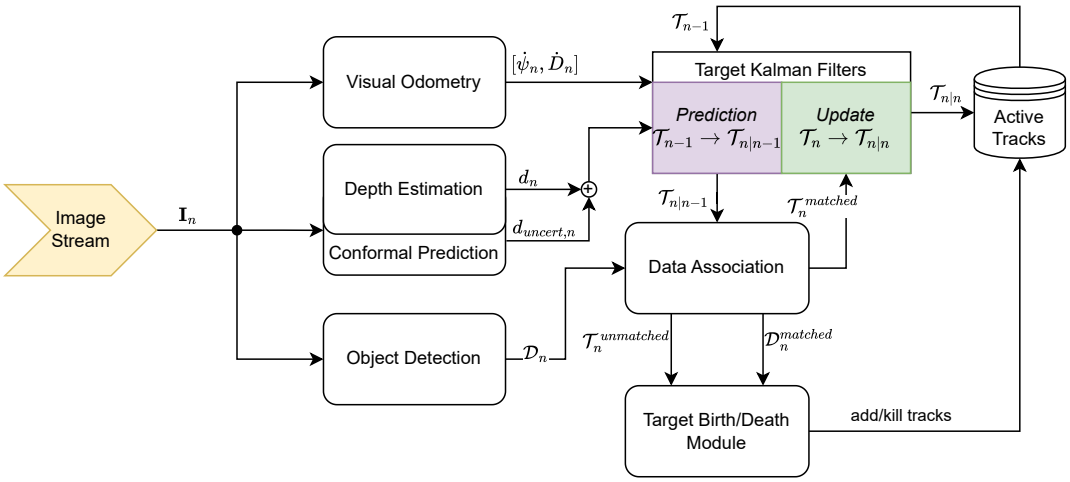


Figure 4.4: Fully vision-based multi-object tracking framework depth uncertainty awareness

within the first few steps

$$\mathbf{P}_0 = \begin{bmatrix} 10\mathbf{R}_0 & 0_{5\times5} \\ 0_{5\times5} & 100\mathbf{R}_0 \end{bmatrix} \tag{4.27}$$

Chapter 5

Experiments and Results

In the following section, the datasets, metrics, baseline trackers, and the results are presented.

5.1 Experimental Setup

5.1.1 KITTI

The KITTI dataset serves as a cornerstone in benchmarking computer vision algorithms for autonomous driving tasks. It consists of real-world data captured from a car equipped with sensors such as LiDAR, camera, and GPS. In the designed experiments, this work uses the KITTI training dataset which consists of 21 sequences of diverse driving scenarios such as urban, highway, and rural environments. For the evaluation, the focus is on the *Car* and *Pedestrian* classes whose bounding boxes are detected by PermaTrack [117] and also YOLOv8 [11].

5.1.2 CARLA Simulation Dataset

In addition to the KITTI dataset, other experiments are conducted using a custom-generated dataset within the CARLA autonomous driving simulator. The CARLA simulator offers a realistic virtual environment for autonomous driving, allowing for controlled and diverse scenarios. Following the previous work [2], I use four sequences each representing distinct features. In Scenario 1, a car navigates a straight road. Scenario 2 involves the car initiating movement from a stationary position, moving along a straight path, encountering a curve, and concluding the sequence upon completing the curve. Scenario 3 features the car starting from a stationary position, progressing along a straight trajectory, taking a left turn, reaching a curve, and coming to a stop afterward. In Scenario 4, the vehicle travels a path with winding movements ending after taking a right turn. Figure 5.1 illustrates the paths on the CARLA *town #10* map. This research runs the MOT algorithms on the curated CARLA dataset through the Robot Operating System (ROS) [118], enabling communication between Python and CARLA. Synchronized messages, including RGB images, depth map (aligned with RGB), ego-vehicle odometry, and automatically generated ground truth tracking bounding boxes from CARLA are streamed to Python via ROS for seamless evaluation.

5.2 Evaluation Metrics

HOTA [119] is selected as the primary metric for the proposed work because of its ability to balance the accuracy of both object detection and association in multi-object tracking. Also, IDSW will be reported as the additional primary metric because it represents the reliability of generated trajectories. The underlying reason is that IDSW counts the occurrences of incorrect object identity swaps or the loss and

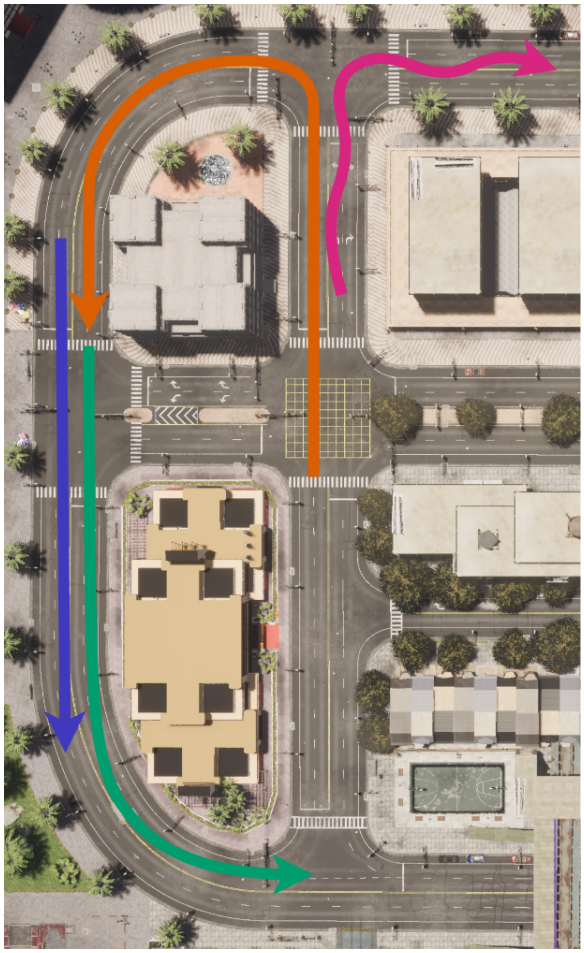


Figure 5.1: Visualization of *town #10* in CARLA simulator, featuring four distinct simulation scenarios superimposed on the map. The paths illustrate the diverse trajectories taken in our dataset, capturing a range of scenarios for comprehensive analysis.

subsequent reinitialization of identities. The analysis of this work extends to Association Accuracy (AssA) for evaluating association performance, along with IDF1 as an additional metric in the same context. It is worth mentioning that MOTA, which used to be the predominant measure of MOT performance, is heavily affected by the detection quality. Hence, to ensure a fair comparison, identical detections among all algorithms are used to report the MOTA score.

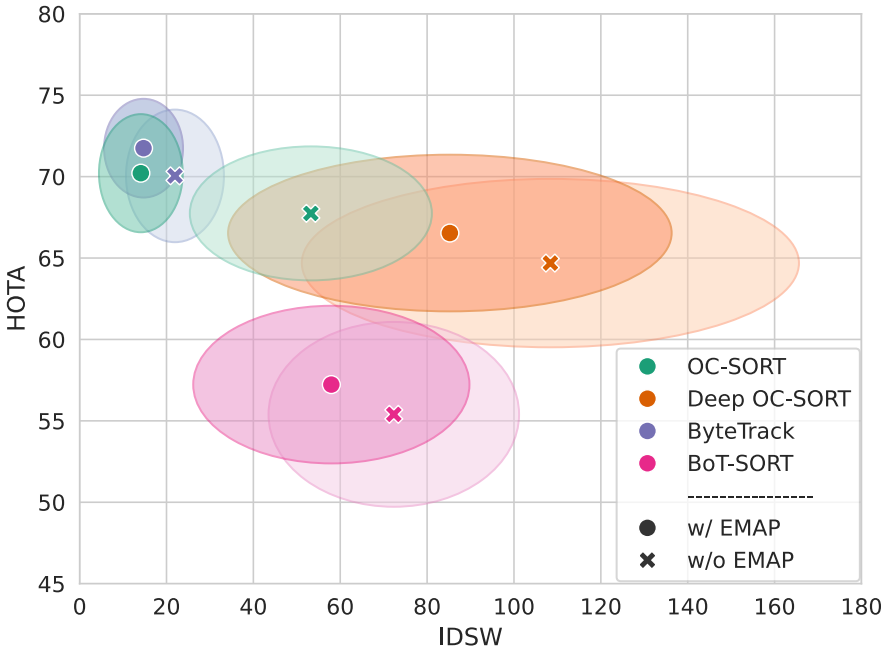


Figure 5.2: HOTA vs IDSW comparisons of OC-SORT, Deep OC-SORT, ByteTrack, and BoT-SORT on 21 KITTI train sequences with or without EMAP module. The height and width of the ellipses are the standard deviation of the distribution.

5.3 Baseline SORT-based Algorithms

In order to evaluate the impact of EMAP on existing MOT algorithms, four state-of-the-art SORT-based algorithms are chosen and integrated with EMAP to evaluate the effect of EMAP on their performance. These core algorithms are: OC-SORT [7], Deep OC-SORT [120], ByteTrack [6], and BoT-SORT [121]. All four baselines are Kalman Filter-based algorithms. However, in BoT-SORT, authors use image registration to approximate the camera motion and compensate for the effect of this motion.

5.4 Results

5.4.1 EMAP

In this section, the performance metrics of the EMAP module on both the CARLA and the KITTI datasets are presented. Table 5.1 displays the HOTA scores and identity switch counts for each sequence in the CARLA simulation dataset. Notably, EMAP significantly impacts Sequence #4 which contains substantial vehicle rotational and translational movements. The performance on this sequence shows a decrease in identity switches and an increase in HOTA values for all trackers, highlighting the power of EMAP in rejecting the harsh ego-motion disturbances.

Table 5.2 presents the average metric values across the 21 sequences of the KITTI train dataset when detections are taken from PermaTrack. EMAP significantly reduces the number of identity switches across all four baselines, with the most substantial impact observed in OC-SORT, where it decreases by 76%. In addition, integrating EMAP improves several metrics, such as HOTA, MOTA, IDF1, FP, FN, AssA, and AssR, in all four baseline trackers. As shown in Figure 5.2, integrating the EMAP module significantly reduces identity switches and enhances HOTA scores compared to baseline performance. It also shows that EMAP enhances the robustness of baseline trackers, as seen in the decreased variance in IDSW and HOTA across all sequences.

In Table 5.3, the evaluation metrics for the KITTI train dataset using YOLOv8x as the object detector are presented. These results illustrate the performance of online object detection. Despite YOLOv8x exhibiting weaker performance compared to PermaTrack, as indicated by higher missed detections and lower detection accuracy, the addition of EMAP still has a positive effect on the performance of all baseline trackers.

When comparing the performance improvement between using YOLOv8 or PermaTrack for detections, it is important to note that the effectiveness of EMAP can still be slightly affected by the quality of the detection. This is because the noise in bounding box detections and odometry readings can lead to disturbed track predictions, especially when detections are lost for an extended period.

Table 5.1: Performance comparison of MOT algorithms with/without EMAP on the curated CARLA dataset split by sequence with YOLOv8x as the object detector. The best results are shown in **bold**.

Tracker	Sequence #1		Sequence #2		Sequence #3		Sequence #4	
	HOTA↑	IDs↓	HOTA↑	IDs↓	HOTA↑	IDs↓	HOTA↑	IDs↓
OC-SORT [7]	36.15	5	24.84	11	36.3	15	24.09	19
OC-SORT + EMAP [7]	**36.24**	**1**	**25.06**	**3**	**37.04**	15	**29.09**	**7**
BoT-SORT [121]	**35.08**	7	21.88	35	**32.64**	30	23.44	24
BoT-SORT + EMAP [121]	34.94	7	**22.14**	35	31.65	30	**24.54**	**18**
Deep OC-SORT [120]	**35.71**	**4**	**23.65**	**55**	**36.33**	**36**	25.17	67
Deep OC-SORT + EMAP [120]	35.55	9	20.75	129	35.55	61	**26.66**	**38**
ByteTrack [6]	34.94	0	21.14	1	34.26	5	25.71	8
ByteTrack + EMAP	**25.05**	0	**21.7**	**0**	**34.91**	**4**	**28.39**	**2**

Table 5.2: Average performance (on 21 sequences) on KITTI-train dataset with/without EMAP, detections are taken from PermaTrack. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
OC-SORT [7]	67.74	64.53	78.24	709.33	234.19	53.24	68.38	79.34
OC-SORT + EMAP	**70.21**	**67.87**	**82.17**	**662.19**	**187.05**	**14.10**	**73.11**	**85.32**
Deep OC-SORT [120]	64.69	62.59	74.20	893.67	418.52	108.38	62.84	74.41
Deep OC-SORT + EMAP	**66.54**	**64.38**	**76.37**	**863.14**	**388.00**	**85.24**	**66.22**	**77.80**
ByteTrack [6]	70.05	74.98	84.60	339.10	305.00	21.95	72.62	80.28
ByteTrack + EMAP	**71.75**	**75.82**	**85.78**	**324.33**	**295.33**	**14.71**	**74.71**	**81.73**
BoT-SORT [121]	55.40	47.74	65.89	602.95	687.29	72.33	56.36	62.33
BoT-SORT + EMAP	**57.23**	**50.11**	**68.49**	**578.29**	**670.76**	**57.95**	**59.58**	**66.05**

5.4.2 VEMAP

By incorporating the unified scale for depth and pose estimations into EMAP, which creates the VEMAP module, the performance of the proposed solution is assessed

Table 5.3: Average performance (on 21 sequences) on KITTI-train dataset with/without EMAP, detections are taken from YOLOv8. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
OC-SORT [7]	39.35	37.0	53.04	163.14	1187.9	18.86	46.16	48.82
OC-SORT + EMAP	**40.28**	**38.0**	**54.4**	168.05	**1179.95**	**12.29**	**47.68**	**50.25**
Deep OC-SORT [120]	38.22	35.77	50.08	270.67	1236.38	69.38	42.21	45.36
Deep OC-SORT + EMAP	**39.6**	**37.2**	**52.88**	**225.48**	**1191.19**	**41.33**	**45.15**	**48.41**
ByteTrack [6]	33.48	31.29	44.66	102.76	1368.76	14.24	42.11	44.2
ByteTrack + EMAP	**33.78**	**31.32**	**45.81**	**83.95**	**1356.81**	**11.9**	**42.69**	44.12
BoT-SORT [121]	25.16	23.5	34.57	138.0	1584.1	46.05	29.17	29.81
BoT-SORT + EMAP	**28.06**	**25.49**	**39.21**	**123.7**	**1425.65**	**35.2**	**33.82**	**34.55**

on the KITTI dataset. Table 5.2 displays the average benchmark values computed over the 21 sequences of the KITTI training dataset when detections are sourced from PermaTrack. VEMAP significantly reduces identity switches compared to four baseline methods, demonstrating its robustness, particularly in OC-SORT, where reductions reach up to 72.5%. The integration of VEMAP has also improved various metrics, including Higher Order Tracking Accuracy (HOTA), Multiple Object Tracking Accuracy (MOTA), ID F1 Score (IDF1), False Positives (FP), False Negatives (FN), Association Accuracy (AssA), and Association Recall (AssR), across all baseline trackers. While VEMAP's enhancements are not as pronounced as those seen with EMAP, they still markedly boost the multi-object tracking (MOT) performance metrics, relying solely on input from a monocular camera without having access to the metric scale factor. the power of VEMAP in making the core MOT method more robust in different environments is also evident by the decline in standard deviation of the HOTA and IDSW metrics in Figure 5.3. Figure 5.4 also illustrates that even when the average MOTA stayed the same after integrating VEMAP, it strengthened the MOT resilience by increasing the repeatability of the MOT methods in various sequences. Table 5.5 also represents the average performance metrics over the 21 KITTI-training dataset when the detections are taken from the YOLOv8 object

detector.

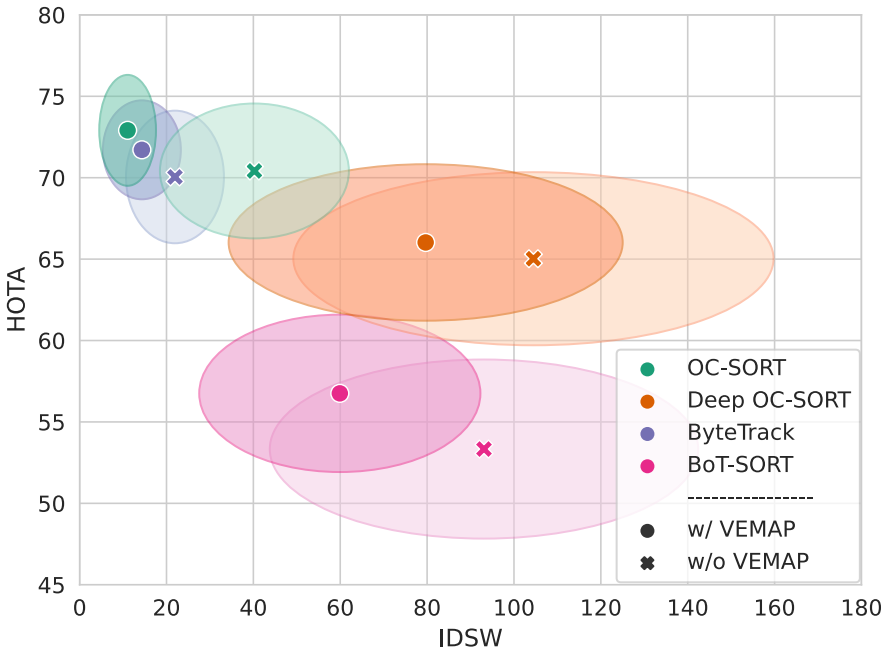


Figure 5.3: HOTA vs IDSW comparisons of OC-SORT, Deep OC-SORT, ByteTrack, and BoT-SORT on 21 KITTI train sequences with or without VEMAP module. The height and width of the ellipses are the standard deviation of the distribution.

Table 5.4: Average performance (on 21 sequences) on KITTI-train dataset with/without VEMAP, detections are taken from PermaTrack. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
OC-SORT [7]	70.41	73.40	81.71	434.19	306.00	40.24	70.18	77.86
OC-SORT + VEMAP	**72.91**	**75.55**	**85.41**	**392.76**	**264.57**	**11.05**	**74.97**	**83.11**
Deep OC-SORT [9]	65.02	62.55	74.38	882.62	407.48	104.48	63.47	74.73
Deep OC-SORT + VEMAP	**66.02**	**64.69**	**76.28**	**843.90**	**368.76**	**79.67**	**65.07**	**77.20**
ByteTrack [6]	70.05	74.98	84.60	339.10	305.00	21.95	72.62	80.28
ByteTrack + VEMAP	**71.71**	**75.95**	**85.76**	**330.57**	**299.95**	**14.33**	**74.58**	**81.76**
BoT-SORT [121]	53.33	45.82	63.08	627.71	759.71	93.10	53.71	59.12
BoT-SORT + VEMAP	**56.76**	**49.73**	**67.79**	**592.48**	**685.19**	**59.90**	**58.74**	**65.41**

Table 5.5: Average performance (on 21 sequences) on KITTI-train dataset with/without VEMAP, detections are taken from YOLOv8. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
OC-SORT [7]	34.1	28.7	43.59	90.7	1340.6	13.25	45.87	48.18
OC-SORT + VEMAP	**34.64**	**29.12**	**44.63**	**83.25**	**1333.15**	**8.70**	**47.1**	**49.27**
Deep OC-SORT [9]	35.34	29.19	44.47	167.4	1309.35	46.8	44.73	47.83
Deep OC-SORT + VEMAP	**36.54**	**30.52**	**46.69**	**135.6**	**1277.55**	**23.85**	**47.63**	**50.72**
ByteTrack [6]	**31.56**	**26.44**	**40.25**	**74.75**	**1408.0**	**10.35**	**45.33**	**47.14**
ByteTrack + VEMAP	29.27	24.9	36.12	129.9	1467.7	12.0	41.13	43.04
BoT-SORT [121]	23.83	19.34	30.53	121.95	1539.75	41.0	29.41	30.24
BoT-SORT + VEMAP	**26.11**	**21.14**	**34.75**	**89.05**	**1493.9**	**20.6**	**33.57**	**34.63**

5.4.3 UVEMAP

After feeding the online depth uncertainty to the Kalman Filter, this data stream is incorporated into VEMAP to evaluate the proposed solution's performance on the KITTI dataset. Table 5.6 displays the average benchmark values computed over the 21 sequences of the KITTI training dataset when detections are sourced from PermaTrack. Table 5.8 also provides the tracker performance on selected sequences with more camera motion and smaller objects. The reduction in the number of identity switches is evident after employing UVEMAP ($\tilde{3}1\%$) demonstrating its effectiveness. This drop is pronounced in *Sequence #5* and *Sequence #13* where camera motion significantly hinders the core tracker's performance. The integration of UVEMAP has also improved various metrics, including Higher Order Tracking Accuracy (HOTA), Multiple Object Tracking Accuracy (MOTA), and False Positives (FP). Table 5.7 also represents the average tracking performance on the KITTI training dataset when detections are sourced from Yolov8. Compared to PermaTrack, YOLOv8 has a significantly higher miss rate which is evident in the number of FN. The evaluation using Yolov8 as the object detector only shows enhancement in terms of dropped identity switches. The poor performance of YOLOv8 compared to PermaTracks, negatively affects the UVEMAP module as the errors in detections propagate over to depth

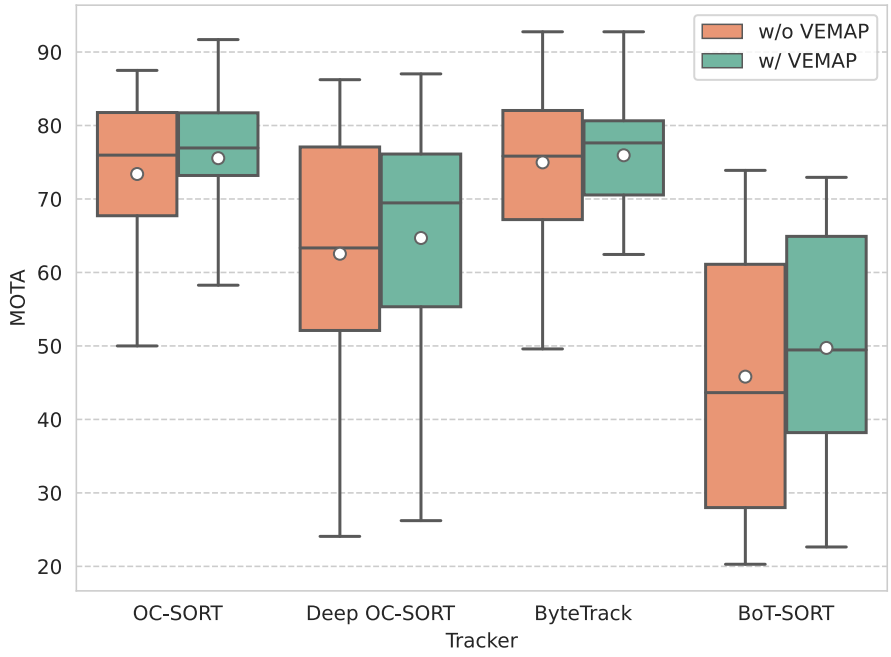


Figure 5.4: Distribution of MOTA metrics over 21 KITTI train sequences with or without VEMAP module. The hollow circle shows the mean of the results while the horizontal line represents the median.

estimation and uncertainty extraction. This showcases the sensitivity of the solution to detection since it only relies on the stream of images.

While UVEMAP's enhancements are not as pronounced as those seen with (V)EMAP, they still markedly boost the Multi-Object Tracking performance metrics, relying solely on input from a monocular camera without any metric information about the 3D world. Figure 5.5 illustrates that integration of the UVEMAP module notably decreases identity switches and enhances HOTA scores compared to baseline performance. Additionally, it demonstrates that UVEMAP contributes to the improved robustness of baseline trackers, as evidenced by the reduced variance in IDSW and HOTA across all sequences.

Table 5.6: Average performance (on 21 sequences) on KITTI-train dataset with/without UVEMAP, detections are taken from PermaTrack. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
ByteTrack [6]	70.05	74.98	**84.60**	339.10	**305.00**	21.95	**72.62**	**80.28**
ByteTrack + UVEMAP	**70.77**	**75.00**	81.39	**328.00**	363.50	**15.15**	71.59	77.47

Table 5.7: Average performance (on 21 sequences) on KITTI-train dataset with/without UVEMAP, detections are taken from Yolo. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
ByteTrack [6]	**34.39**	**28.95**	**46.07**	**114.86**	**1223.81**	13.95	**43.21**	**45.53**
ByteTrack + UVEMAP	32.76	27.82	41.94	167.95	1285.05	**12.71**	39.22	40.82

Table 5.8: Performance comparison of UVEMAP on selected sequences of KITTI-train.

Sequence	ByteTrack				ByteTrack + UVEMAP			
	HOTA↑	MOTA↑	IDs↓	IDF1↑	HOTA↑	MOTA↑	IDs↓	IDF1↑
Sequence #0	64.37	64.09	1	80.23	69.24	66.02	0	83.39
Sequence #1	67.10	75.83	45	83.49	74.77	76.04	11	86.010
Sequence #4	65.42	71.62	47	80.76	63.48	73.39	34	68.56
Sequence #5	73.43	79.28	28	88.12	76.55	78.17	5	86.03
Sequence #9	66.69	71.81	44	83.74	69.49	70.95	29	82.23
Sequence #11	76.59	81.08	25	88.63	71.72	78.78	35	77.95
Sequence #13	48.92	49.59	52	64.57	68.81	59.18	12	77.91
Sequence #15	65.95	71.43	15	78.07	63.73	68.49	17	73.27
Sequence #19	58.68	67.19	79	74.09	57.21	66.84	68	68.66

5.5 Visualization

5.5.1 Depth Estimation and Conformal Prediction

In this work, Monodepth2 was utilized after implementing the necessary modifications to make it compatible with the proposed solution. Representative results from the KITTI dataset are displayed in Figure 5.6, showcasing the estimated depth. It illustrates the prediction interval of the estimated depth computed using split conformal prediction.

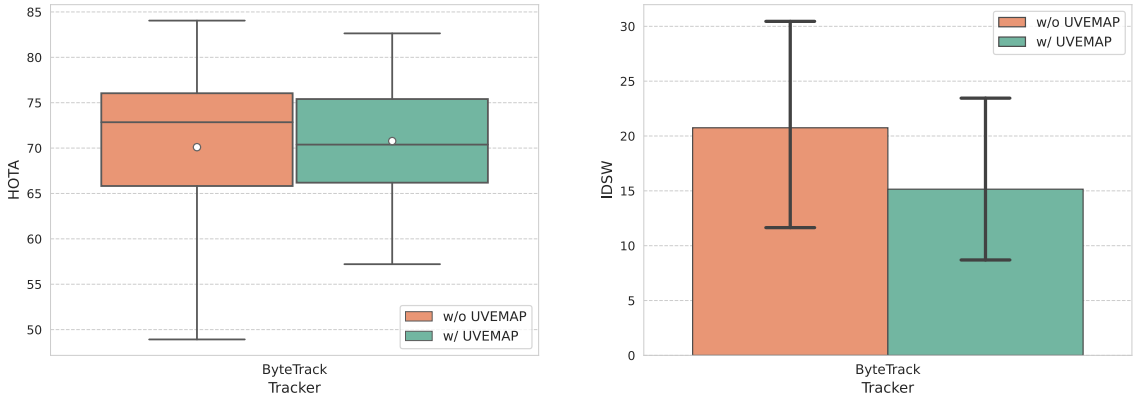


Figure 5.5: Distribution of HOTA and IDSW metrics over 21 KITTI train sequences with or without UVEMAP module on ByteTrack. The hollow circle shows the mean of the results while the horizontal line represents the median.

5.5.2 Visual Odometry

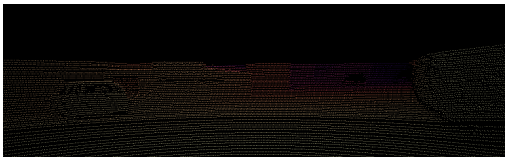
I compare ground truth trajectories with those predicted by the pose network, emphasizing the precision and reliability of Monodepth2 estimations. Figure 5.7 illustrates the estimated trajectory by Monodepth2 over two sample sequences of the KITTI training dataset.

This visual representation vividly showcases the resilience of the proposed solution in coping with significantly imprecise estimated trajectories. The standout feature of (U)VEMAP lies in its robustness in addressing inaccuracies in visual odometry while enhancing the baseline performance to align with that of EMAP.

5.5.3 EMAP

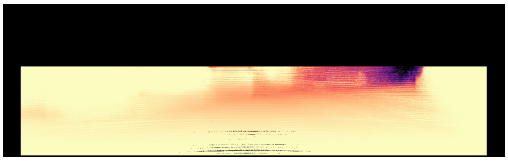
To showcase the impact of EMAP module, a visualization is provided to depict the performance of the Bytetrack MOT algorithm operating within the CARLA simulator under a scenario where detection is lost. As illustated in Figure 5.8 (a), the prediction module of the vanilla ByteTrack struggles to accurately anticipate the position of the

(a) Image instance

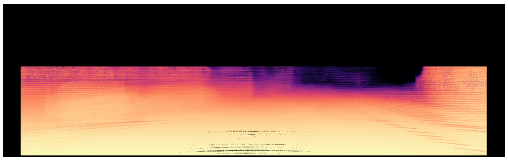


(b) Depth ground truth

(c) Estimated Depth



(d) Depth lower bound



(e) Depth upper bound

Figure 5.6: An instance of KITTI images and the extracted depth and its boundary using Monodepth2 and split conformal prediction. Lighter shades indicate closer objects while darker shades represent higher depth.

target after the detection becomes unavailable (denoted by black bounding boxes). In contrast, Figure 5.8 (b) illustrates the enhanced performance achieved by incorporating the EMAP module. The predicted bounding box location aligns more closely with the actual location, underscoring the efficacy of the ego-motion aware prediction module.

5.5.4 UVEMAP

The proposed modification of the Kalman filter contributes to improving the fit of the bounding box width to the object, as can be seen in figure 5.9. It showcases some of the core differences when UVEMAP is added to ByteTrack. In the first instance, the core tracker loses the *person* track due to the vehicle's translational movement. The

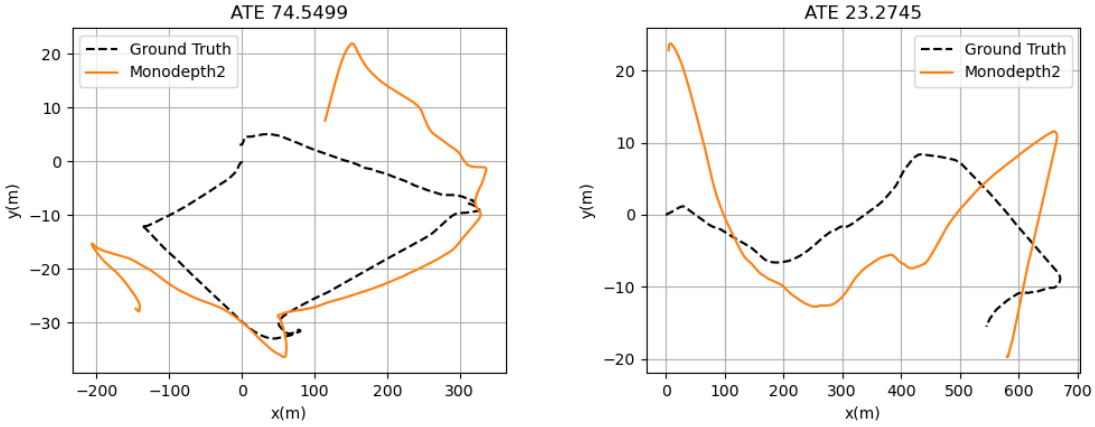


Figure 5.7: Instances of trajectory estimation using Monodepth2 and monocular camera stream of KITTI training set.

second example demonstrates the effectiveness of UVEMAP in reducing false tracks, and the last instance illustrates how it eliminates the lag in tracking the bounding boxes. Additionally, the number of IDs is significantly lower in all the representative results. This matter is also quantitatively shown by the remarkable drop in the number of identity switches.

(a) (b)

Figure 5.8: In this scenario, the object detection is missed, leading to failure in predicting the location of the object during lane-changing for Vanilla ByteTrack (a). ByteTrack + EMAP (b) significantly improves the prediction, successfully tracking the object in the next two frames.

(a) ByteTrack (b) ByteTrack + UVEMAP

Figure 5.9: Instances where the core ByteTrack fails to precisely locate the object track or completely misses it. The addiction of UVEMAP enhances the overall performance of the core tracker

Chapter 6

Conclusions and Future Work

6.1 Summary

Throughout this thesis, the UVEMAP system was developed, a vision-based ego-motion-aware target prediction module with uncertainty awareness designed for robust multi-object tracking using only monocular camera inputs. This research aimed to enhance the performance of multi-object tracking systems by integrating ego-motion awareness extracted from pure image sources and quantifying the uncertainty in depth estimation.

The project began by decoupling the effect of camera motion in Kalman-based multi-object tracking algorithms. This step was crucial in improving the accuracy of tracking by ensuring that the motion of the ego-vehicle did not adversely affect the predicted locations of objects. Building on this foundation, this work was extended [2] by eliminating the need for additional sensory data such as IMUs and GPS. By focusing on a pure vision-based solution that is agnostic of depth true scale, I developed a system that relies solely on the image stream from a monocular camera. This approach not only reduced the dependency on external sensors but also enhanced the

versatility and applicability of the system across different domains.

A significant aspect of this research was the integration of depth estimation and visual odometry using Monodepth2. This method employs a self-supervisory signal derived from image reprojection errors to estimate depth and camera motion. The modifications made to Monodepth2 ensured compatibility with our solution, enabling accurate depth and pose estimation from monocular images. By leveraging these estimates, The proposed solution was able to accurately predict object locations and account for the motion of the ego-vehicle.

Furthermore, UVEMAP introduced conformal prediction techniques to quantify the uncertainty in depth estimation. By calculating pixel-wise nonconformity scores in the dataset, prediction intervals were determined to provide a measure of the uncertainty associated with each depth estimate. This information was then integrated into the Kalman Filter, enabling it to acquire real-time noise measurement.

The successful integration of unified scale depth and pose estimations, along with depth uncertainty quantification, into the EMAP framework significantly enhanced the performance metrics across the baseline method. The method exploits real-time data on measurement noise, enabling more precise and reliable object tracking under various operational conditions.

One of the key strengths of the proposed approach is its computational efficiency. By employing heuristic methods, UVEMAP ensures that the computational load is minimized, making it highly suitable for deployment on edge devices. My review of model compression methods also serves as a guideline for compression neural networks where such efficient deployment is desired. The computational efficiency, combined with the robustness and accuracy of the system, makes UVEMAP an ideal solution for real-world applications where processing power and resources are limited.

The incorporation of these advancements into UVEMAP has been demonstrated

to enhance the robustness of the core tracker without requiring any external sensory data or metric data streams. By relying solely on monocular camera inputs, UVEMAP provides a cost-effective and scalable solution for robust multi-object tracking.

In conclusion, the development of UVEMAP marks a significant step forward in the advancement of multi-object tracking systems. The integration of vision-based ego-motion estimation and depth uncertainty quantification has been demonstrated to be effective in enhancing tracking accuracy and robustness. Future work may include, but is not limited to, the extension of this framework to incorporate additional sources of information, such as semantic segmentation, to further improve the system's performance in complex and dynamic environments.

6.2 Limitations

Despite the significant advancements and promising results achieved with the UVEMAP system, several limitations remain that could impact its performance and applicability in certain scenarios. Addressing these limitations in future work will be crucial for further enhancing the robustness and accuracy of the system.

Dependency on Quality of Monocular Images. The UVEMAP system relies solely on monocular camera inputs for depth estimation and visual odometry. While this approach reduces the dependency on additional sensory data, it also means that the quality and resolution of the monocular images are critical to the system's performance. In low-light conditions or environments with significant visual noise, the accuracy of depth estimation and ego-motion prediction may be compromised. Additionally, rapid changes in lighting or extreme weather conditions can further degrade the quality of the input images, impacting the overall tracking performance.

Handling Dynamic and Crowded Environments. While UVEMAP has demonstrated robust performance in various scenarios, tracking multiple objects in highly dynamic and crowded environments remains a challenge. The presence of numerous occlusions, interactions between objects, and rapid movements can lead to increased identity switches and tracking errors. Although the conformal prediction approach helps quantify uncertainty and improve robustness, further enhancements are needed to effectively manage such complex scenarios.

Scale Ambiguity in Depth Estimation. The depth estimation component of UVEMAP is based on monocular images, which inherently lack absolute scale information. While the system can estimate relative depth and motion accurately, it cannot determine the exact metric distances without additional scale information. This scale ambiguity can limit the applicability of UVEMAP in scenarios where precise metric measurements are required. Although the current implementation mitigates this limitation by being agnostic to the metric scale, integrating external sources of scale information or developing novel techniques to infer scale from monocular images would provide UVEMAP module with absolute scale uncertainty information.

Computational Overhead. Despite the heuristic nature of the UVEMAP approach, which ensures computational efficiency, the integration of depth estimation, visual odometry, and uncertainty quantification introduces some computational overhead. Deploying the system on edge devices with limited processing power, such as autonomous ground robots and vehicles, may require further optimization to ensure real-time performance. Balancing computational efficiency with the accuracy and robustness of the tracking system remains an ongoing challenge.

The limitations identified in this research provide valuable insights and directions for future work. Enhancing the robustness of the UVEMAP system to handle diverse

environmental conditions, dynamic and crowded scenes, object-with-objects interaction, and scale ambiguity will be crucial. Additionally, optimizing the computational efficiency and validating the system on a wider range of datasets will help to improve its practical applicability and generalization capabilities.

6.3 Future Work

Looking ahead, there are several potential directions for further research. One promising avenue is the integration of additional visual cues, such as semantic segmentation, to provide richer contextual information that can improve tracking performance. Using semantic masks to extract depth from the scene will remove the unnecessary background pixel from the target which in return enhances the accuracy of the extracted depth.

Moreover, expanding the evaluation of the UVEMAP system to include more diverse and challenging datasets, will help validate its robustness and generalizability across different scenarios.

Another area of interest is the adaptation of other variants of Kalman Filter. Although employing the original Kalman Filter due to its simplicity and adaptability favors the (UV)EMAP solution, in cases where target motion violates the constant velocity assumption, using the Fading Memory Kalman Filter might be beneficial. This variant of the Kalman Filter puts more emphasis on recent observations and discounts the error in older measurements. That is, when the system behavior deviates from the model, using this filter reduces the lag in the estimation. However, one should consider that utilizing this filter requires extra parameter tuning, namely the forgetting factor, which would make the filter design more sophisticated. Also, utilizing an adaptive Kalman filter which can tune itself based on the environment

recognition can compensate for the disturbances caused by different target velocities [122]. For instance, in an indoor environment with slow objects, the filter would adapt to more stable detections by changes made in the process noise matrix.

Additionally, considering a motion model for the ego-vehicle, which may or may not have interaction with other objects in the environment, could potentially reduce the disturbances in target state estimation. Such implementation is illustrated in Figure 6.1 showcasing how the modularity of the (UV)EMAP method makes it suitable for adding extra sub-modules, such as ego-vehicle Kalman Filter, in the scenario mentioned above.

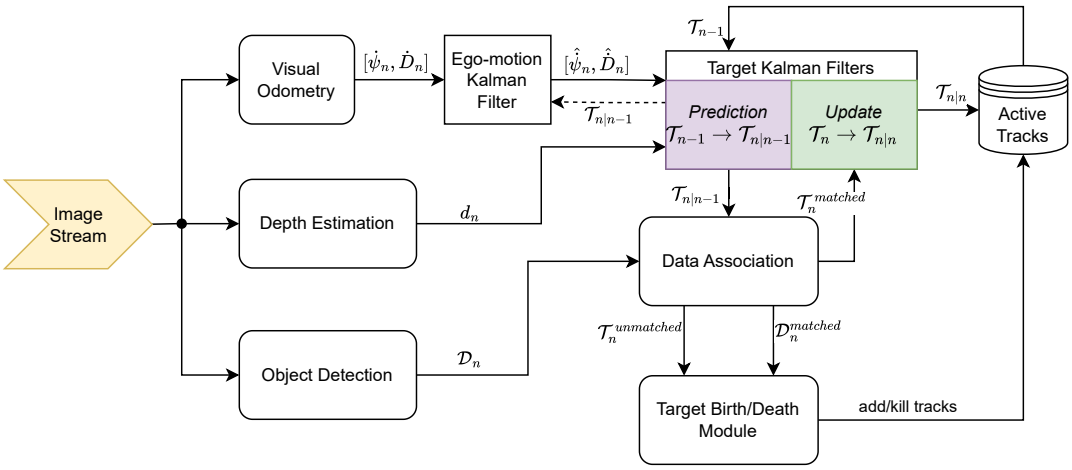


Figure 6.1: One possible framework for future work containing a Kalman Filter block for the ego-vehicle separated from the targets' Kalman Filters

By pursuing these future research directions, one can continue to advance the field of multi-object tracking and develop more robust, accurate, and versatile tracking systems that can meet the demands of a wide range of applications.

Appendix A

Ablation Study

To assess the individual and combined impact of the translation and rotational submodules, we conduct an ablation study comparing the results of EMAP with each submodule added separately and together, as shown in Table A.1. The largest reduction in the number of identity switches occurs when both submodules are active. However, the maximum HOTA is attained when only the rotational module is activated.

Table A.1: Ablation on KITTI dataset with Translational and Rotational submodules. The best results are shown in **bold**.

Method	Tracker	HOTA↑	MOTA↑	IDF1↑	IDs↓
Baseline	OC-SORT	67.74	64.53	78.24	53.24
	Deep OC-SORT	64.69	62.59	74.20	108.38
	ByteTrack	70.05	74.98	84.60	21.95
	BoT-SORT	55.40	47.74	65.89	72.33
Translational-Only EMAP	OC-SORT	71.45	74.59	83.35	24.38
	Deep OC-SORT	65.33	63.08	74.91	99.43
	ByteTrack	70.25	74.77	84.40	23.00
	BoT-SORT	55.28	48.65	65.58	76.71
Rotational-Only EMAP	OC-SORT	**72.63**	75.10	85.05	15.43
	Deep OC-SORT	65.57	63.72	75.62	97.14
	ByteTrack	71.18	**75.86**	85.69	15.10
	BoT-SORT	55.41	48.65	65.59	85.81
EMAP	OC-SORT	70.21	67.87	82.17	**14.1**
	Deep OC-SORT	66.54	64.38	76.37	85.24
	ByteTrack	71.75	75.82	**85.78**	14.71
	BoT-SORT	57.23	50.11	68.49	57.95

Bibliography

[1] Mohammad Jani, Jamil Fayyad, Younes Al-Younes, and Homayoun Najjaran. Model compression methods for yolov5: A review. *arXiv preprint arXiv:2307.11904*, 2023.

[2] Navid Mahdian, Mohammad Jani, Amir M Soufi Enayati, and Homayoun Najjaran. Ego-motion aware target prediction module for robust multi-object tracking. *arXiv preprint arXiv:2404.03110*, 2024.

[3] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[4] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 941–951, 2019.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, September 2016. ISSN: 2381-8549.

[6] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-object Tracking by Associating Every Detection Box. In *Computer Vision – ECCV 2022: 17th*

European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII, pages 1–21, Berlin, Heidelberg, October 2022. Springer-Verlag.

[7] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9686–9696, Vancouver, BC, Canada, June 2023. IEEE.

[8] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. StrongSORT: Make DeepSORT Great Again, February 2023. arXiv:2202.13514 [cs].

[9] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric, March 2017. arXiv:1703.07402 [cs].

[10] Shoudong Han, Piao Huang, Hongwei Wang, En Yu, Donghaisheng Liu, and Xiaofeng Pan. MAT: Motion-aware multi-object tracking. *Neurocomputing*, 476(C):75–86, March 2022.

[11] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[12] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[14] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. In *2017*

IEEE International Conference on Computer Vision (ICCV), pages 300–311, Venice, October 2017. IEEE.

[15] Anton Milan, S. Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 4225–4232, San Francisco, California, USA, February 2017. AAAI Press.

[16] Nathanael L. Baisa. Occlusion-robust online multi-object visual tracking using a GM-PHD filter with CNN-based re-identification. *Journal of Visual Communication and Image Representation*, 80:103279, October 2021.

[17] Qiankun Liu, Dongdong Chen, Qi Chu, Lu Yuan, Bin Liu, Lei Zhang, and Nenghai Yu. Online multi-object tracking with unsupervised re-identification learning and occlusion estimation. *Neurocomputing*, 483:333–347, April 2022.

[18] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. Mars: A video benchmark for large-scale person re-identification. volume 9910, pages 868–884, 10 2016.

[19] B.-N. Vo and W.-K. Ma. The Gaussian mixture probability hypothesis density filter. *IEEE Transactions on Signal Processing*, 54(11):4091–4104, 2006.

[20] Aria Salari, Abtin Djavadifar, Xiangrui Liu, and Homayoun Najjaran. Object recognition datasets and challenges: A review. *Neurocomputing*, 495:129–152, 2022.

[21] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020.

[22] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

[23] Silvia González, Javier Sedano, José R Villar, Emilio Corchado, Álvaro Herrero, and Bruno Baruque. Features and models for human activity recognition. *Neurocomputing*, 167:52–60, 2015.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[25] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[26] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[29] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30:3212–3232, 2019.

[30] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019.

[31] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29, 2016.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[33] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[34] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[35] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[36] Glenn Jocher. Ultralytics yolov5, 2020.

[37] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[38] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[39] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.

[40] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[41] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

[42] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. PMLR, 2015.

[43] Jian-Xun Mi, Jie Feng, and Ke-Yang Huang. Designing efficient convolutional neural network structure: A survey. *Neurocomputing*, 489:139–156, 2022.

[44] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[45] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[46] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[47] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[48] Babak Hassibi, David Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. *Advances in neural information processing systems*, 6, 1993.

[49] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 11 2016.

[50] Yanqi Chen, Zhengyu Ma, Wei Fang, Xiawu Zheng, Zhaofei Yu, and Yonghong Tian. A unified framework for soft threshold pruning. *arXiv preprint arXiv:2302.13019*, 2023.

[51] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 662–677. Springer, 2016.

[52] Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

[53] Sheng Xu, Anran Huang, Lei Chen, and Baochang Zhang. Convolutional neural network pruning: A survey. In *2020 39th Chinese Control Conference (CCC)*, pages 7458–7463. IEEE, 2020.

[54] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[55] Kaveena Persand, Andrew Anderson, and David Gregg. Composition of saliency metrics for pruning with a myopic oracle. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 753–759. IEEE, 2020.

[56] Bin Dai, Chen Zhu, Baining Guo, and David Wipf. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, pages 1135–1144. PMLR, 2018.

[57] Ning Liu, Xiaolong Ma, Zhiyuan Xu, Yanzhi Wang, Jian Tang, and Jieping Ye. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 04, pages 4876–4883, 2020.

[58] Xiaolong Ma, Sheng Lin, Shaokai Ye, Zhezhi He, Linfeng Zhang, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, et al. Non-structured dnn weight pruning—is it beneficial in any platform? *IEEE transactions on neural networks and learning systems*, 33(9):4930–4944, 2021.

[59] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005, 2021.

[60] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very

deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[61] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *arXiv preprint arXiv:2001.08565*, 2020.

[62] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, 13(3), feb 2017.

[63] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. *Advances in Neural Information Processing Systems*, 34:19637–19651, 2021.

[64] Benedetta Liberatori, Ciro Antonio Mami, Giovanni Santacatterina, Marco Zullich, and Felice Andrea Pellegrino. Yolo-based face mask detection on low-end devices using pruning and quantization. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 900–905, 2022.

[65] Chengcheng Li, Zi Wang, Xiangyang Wang, and Hairong Qi. Single-shot channel pruning based on alternating direction method of multipliers. *arXiv preprint arXiv:1902.06382*, 2019.

[66] Tuanjie Shao and Dongkun Shin. Structured pruning for deep convolutional neural networks via adaptive sparsity regularization. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 982–987. IEEE, 2022.

[67] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

[68] Fangcheng Fu, Yuzheng Hu, Yihan He, Jiawei Jiang, Yingxia Shao, Ce Zhang, and Bin Cui. Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript. In *International Conference on Machine Learning*, pages 3304–3314, 2020.

[69] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019.

[70] Chaim Baskin, Natan Liss, Eli Schwartz, Evgenii Zheltonozhskii, Raja Giryes, Alex M. Bronstein, and Avi Mendelson. Uniq: Uniform noise injection for non-uniform qantization of neural networks. *ACM Transactions on Computer Systems*, 37(1-4), 2021.

[71] Kohei Yamamoto. Learnable companding quantization for accurate low-bit neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5029–5038, 2021.

[72] Olivia Weng. Neural network quantization for efficient inference: A survey. *arXiv preprint arXiv:2112.06126*, 2021.

[73] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11-12):3245–3262, 2021.

[74] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the Connectivity: Multi-Object Tracking with TrackletNet. In

Proceedings of the 27th ACM International Conference on Multimedia, pages 482–490, Nice France, October 2019. ACM.

[75] Haodong Liu, Tianyang Xu, and Xiaojun Wu. MMOT: Motion-Aware Multi-Object Tracking with Optical Flow. In *Proceedings of the 2022 11th International Conference on Computing and Pattern Recognition*, pages 115–120, Beijing China, November 2022. ACM.

[76] Bing Shuai, Andrew Berneshawi, Xinyu Li, Davide Modolo, and Joseph Tighe. SiamMOT: Siamese Multi-Object Tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12367–12377, Nashville, TN, USA, June 2021. IEEE.

[77] Kuan-Chih Huang, Ming-Hsuan Yang, and Yi-Hsuan Tsai. Delving into motion-aware matching for monocular 3d object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6909–6918, 2023.

[78] Georgios D. Evangelidis and Emmanouil Z. Psarakis. Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, October 2008. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[79] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *Conference on Robot Learning*, pages 34–45. PMLR, 2022.

[80] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsuper-

vised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017.

[81] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. In *Conference on Robot Learning*, pages 1761–1772. PMLR, 2021.

[82] Shihao Shen, Yilin Cai, Wenshan Wang, and Sebastian Scherer. Dytanvo: Joint refinement of visual odometry and motion segmentation in dynamic environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4048–4055. IEEE, 2023.

[83] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 340–349, 2018.

[84] Kuan Xu, Yuefan Hao, Shenghai Yuan, Chen Wang, and Lihua Xie. Airvo: An illumination-robust point-line visual odometry. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3429–3436. IEEE, 2023.

[85] Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, and Ian Reid. Visual odometry revisited: What should be learnt? In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 4203–4210. IEEE, 2020.

[86] Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, Ravi Garg, and Ian Reid. Df-vo: What should be learnt for visual odometry? *arXiv preprint arXiv:2103.00933*, 2021.

[87] Ue-Hwan Kim, Se-Ho Kim, and Jong-Hwan Kim. Simvodis: Simultaneous visual odometry, object detection, and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):428–441, 2020.

[88] André O Françani and Marcos ROA Maximo. Transformer-based model for monocular visual odometry: a video understanding approach. *arXiv preprint arXiv:2305.06121*, 2023.

[89] Fangwen Shu, Jiaxuan Wang, Alain Pagani, and Didier Stricker. Structure plp-slam: Efficient sparse mapping and localization using point, line and plane for monocular, rgb-d and stereo cameras. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2105–2112. IEEE, 2023.

[90] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017.

[91] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019.

[92] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8001–8008, 2019.

[93] Jia-Wang Bian, Huangying Zhan, Naiyan Wang, Zhichao Li, Le Zhang, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent

depth learning from video. *International Journal of Computer Vision*, 129(9):2548–2564, June 2021.

[94] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. *Advances in neural information processing systems*, 32, 2019.

[95] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2022–2030, 2018.

[96] Seokju Lee, Sunghoon Im, Stephen Lin, and In So Kweon. Learning monocular depth in dynamic scenes via instance-aware projection consistency. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 1863–1872, 2021.

[97] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1611–1621, 2021.

[98] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[99] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023.

[100] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. *arXiv preprint arXiv:1812.04605*, 2018.

[101] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (ToG)*, 39(4):71–1, 2020.

[102] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*, 2024.

[103] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019.

[104] Jeffrey Humpherys, Preston Redd, and Jeremy West. A fresh look at the kalman filter. *SIAM review*, 54(4):801–823, 2012.

[105] James B Rawlings and Bhavik R Bakshi. Particle filtering and moving horizon estimation. *Computers & chemical engineering*, 30(10-12):1529–1541, 2006.

[106] Wenling Li and Yingmin Jia. H-infinity filtering for a class of nonlinear discrete-time systems based on unscented transform. *Signal Processing*, 90(12):3301–3307, 2010.

[107] Nicholas Assimakis, Maria Adam, and Anargyros Douladiris. Information filter and kalman filter comparison: Selection of the faster filter. In *Information Engineering*, volume 2, pages 1–5, 2012.

[108] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in neural information processing systems*, 33:14927–14937, 2020.

[109] Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

[110] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine learning: ECML 2002: 13th European conference on machine learning Helsinki, Finland, August 19–23, 2002 proceedings 13*, pages 345–356. Springer, 2002.

[111] Jing Lei and Larry Wasserman. Distribution-free prediction bands for nonparametric regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 76(1):71–96, 2014.

[112] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[113] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[114] Volodya Vovk, Alexander Gammerman, and Craig Saunders. Machine-learning applications of algorithmic randomness. 1999.

[115] Amr Alkhatib, Henrik Bostrom, Sofiane Ennadir, and Ulf Johansson. Approximating score-based explanation techniques using conformal regression. In *Conformal and Probabilistic Prediction with Applications*, pages 450–469. PMLR, 2023.

[116] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[117] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10860–10869, 2021.

[118] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *IEEE International Conference on Robotics and Automation Workshop on Open Source Software*, 2009.

[119] Jonathon Luiten, Aljossa Ossep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixe, and Bastian Leibe. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision*, 129(2):548–578, February 2021.

[120] Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, and Kris Kitani. Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification, February 2023. arXiv:2302.11813 [cs].

[121] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking, July 2022. arXiv:2206.14651 [cs].

[122] Homayoun Najjaran and Andrew Goldenberg. Real-time motion planning of an autonomous mobile manipulator using a fuzzy adaptive kalman filter. *Robotics and Autonomous Systems*, 55(2):96–106, 2007.