Single-Eye View: Monocular Real-time Perception Package for Autonomous Driving

Aiyinsi Zuo*[1], Zirui Li*[1], Haixi Zhang*[1], Chunshu Wu[1], Tong Geng[2], Zhiyao Duan[2]

***Abstract*— Amidst the ascendency of camera based autonomous driving technology, sometimes effectiveness is overally focused with limited attention to computational demand. To address these issues, this paper introduces LRHPerception, a real-time monocular perception package for autonomous driving that uses single view monocular camera videos to produce interpretation of the surroundings. It innovatively fuses the computational efficiency of end-to-end learning with the comprehensive details intrinsic to local mapping methodologies: With significant enhancements across object tracking and prediction, road segmentation, and depth estimation encapsulated in a cohesive system, LRHPerception proficiently processes monocular image data, yielding a five-channel tensor comprising the original RGB, road segmentation, and pixel-level depth estimation channels, embellished with object detection and trajectory prediction. Empirical evaluations substantiate its superior performance, showcasing a single-GPU real-time processing rate of 29 FPS, a 555% acceleration over the fastest mapping technique, attributed equally to modular enhancements and our unique amalgamation technique. The code is available at LRHPerception.**

Index Terms— Computer Vision for Transportation, Deep Learning for Visual Perception, Visual Recognition

I. Introduction

In recent years, the domain of autonomous driving has witnessed significant progress, particularly in cost-effective camera-based technologies. Such research is also driven by the aspiration to mirror human driving, ultimately leading to safer and more understandable vehicles [1]. Predominantly, two methodologies have emerged: 1) end-to-end neural networks that interpret raw images to directly produce steering commands [2], [3], and 2) multi-camera fusion techniques that generate a bird's eye view or 3D occupancy map for path planning [4]–[6]. While groundbreaking, they have their respective constraints.

End-to-end learning methods transform raw imagery into driving decisions, lauded for their computational prowess [3]. Yet, they often face scrutiny due to limited interpretability and unpredictability from simplified input processing design [7]. These systems, despite performing admirably under training-similar conditions, might falter in unfamiliar, dynamic traffic scenarios [8], potentially leading to unsafe decisions. Conversely, systems utilizing imagery from multiple cameras provide an encompassing view of the environment, aiding technicians in comprehension and system enhancement. Their computational demands, however, can sometimes impede real-time processing on standard hardware with single GPU, limiting practicality in live scenarios.

Our paper presents a novel monocular real-time perception system for autonomous driving, addressing key challenges in this domain. Traditional approaches have extensively researched individual areas like road detection, pixel-depth estimation, object detection, and trajectory predictions. However, these domains have often been studied in isolation. We bridge this gap by not only enhancing each module but also introducing novel integration techniques that seamlessly fuse these components. By processing single-camera video feeds, our system, LRHPerception (**L**ow-cost, **R**eal-time, **H**igh Information richness), offers a robust blend of resource efficiency with an information-rich perception of the driving scenario.

LRHPerception processes RGB images, providing road segmentation, pixel-depth estimation, object detection, and trajectory predictions. Unique modules for each task integrate computationally efficient blocks and structures, ensuring comparable or superior accuracy. Integration involves shared backbones and skip feature map connections, reducing repetitive input data processing. This pioneering effort represents the first instance of amalgamating these modules into a comprehensive package. Our key contributions include:

1) We introduce "LRHPerception", a unified and pragmatic approach to autonomous perception that efficiently implements object tracking, trajectory prediction, road segmentation, and depth estimation, all derived from a monocular camera's video input. To our knowledge, this is the first work to integrate these modules into a cohesive package for real-time processing.

2) We implement substantial innovation across each module of monocular image perception, consistently surpassing contemporary state-of-the-art benchmarks. These advancements not only contribute to faster processing speeds but also achieve either comparable or superior perception accuracy.

3) We present an integration technique that consolidates all modules within the package, facilitating information sharing to reduce redundant processing. This integration, combined with module-specific innovations, achieves a 555% acceleration compared to the fastest local-mapping method.

4) The LRHPerception package constitutes the first block in the robotic pipeline of "Perception-Cognition-Action" under our vision of creating a practical and efficient monocular-camera-based autonomous driving system.

II. Related Works

A. End-to-End Training

Contrary to traditional methods, which segregate processes like localization and mapping, planning, and control [9], end-to-end algorithms seek to unify these processes into a single learned model, directly translating raw sensory data into output control commands. This idea was first exemplified by the

*Equal Contribution

[1]Aiyinsi Zuo, Zirui Li, Haixi Zhang, and Chunshu Wu are with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627, USA
{azuo, zli133, hzh104}@u.rochester.edu, cwu88@ur.rochester.edu

[2]Zhiyao Duan and Tong Geng are with the Faculty of the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627, USA
{zhiyao.duan,tong.geng}@rochester.edu

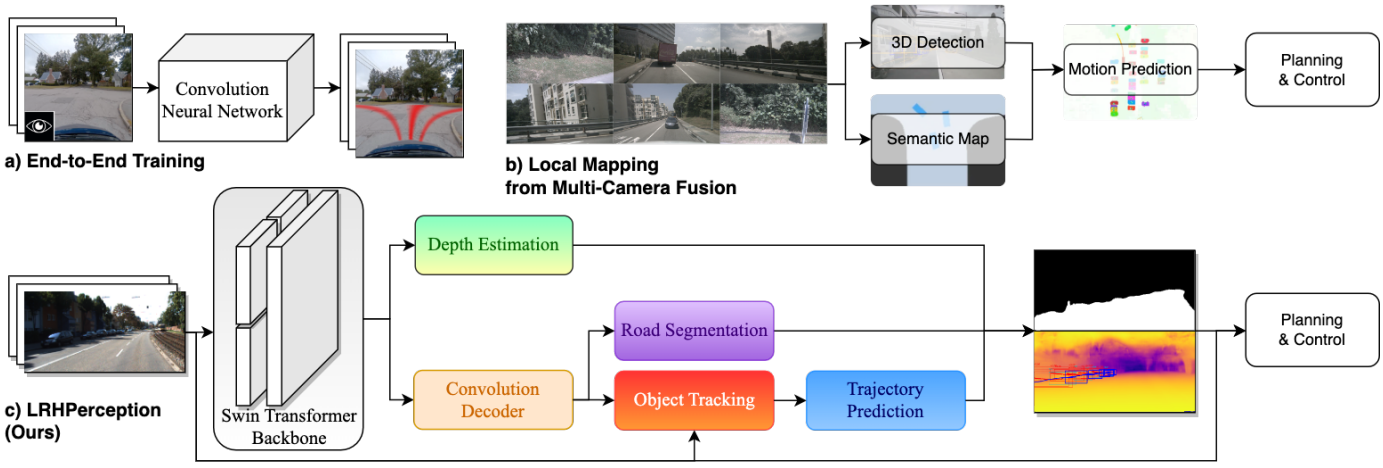

Fig. 1: **Innovation and architecture blueprint** a) Paradigm of end-to-end solution b) Paradigm of camera-fusion for local map solution c) Paradigm of our LRHPerception package, extracts essences from monocular camera for cost-info trade-off.

ALVINN system [10], which employed a multilayer perceptron to learn the vehicle's steering direction. With the rise of convolutional neural networks (CNNs), the ability to learn deterministic [2] or probabilistic [11] driving commands from raw imagery has significantly evoloved, enabling sustained driving [8] and intricate lane-change maneuvers [12].

Existing models often struggle to address the inherent ambiguity, commonly described as the "black-box nature," [7] in yielding steering possibilities, which can create obscure limitations and pose a potential safety risk. Moreover, they frequently neglect the importance of considering interactions with other traffic participants [10], [11], which can hinder performance, especially in complex and dynamic traffic scenarios.

B. Local Mapping via Multi-Camera Fusion

The fusion of multi-camera data to enable simultaneous localization and mapping (SLAM) and thus construct and update maps of uncharted environments, has attracted substantial research attention [4], [6], [13]. Its capacity to furnish a panoramic view of the surroundings enriches trajectory prediction and distance computation: Numerous methodologies have been proffered to accomplish multi-camera fusion to link with downstream tasks of object tracking and prediction [14], [15]. Despite their remarkable results, the inherent complexity of the tasks precludes deployment on a singular, unified system [13]. Specifically, current state-of-the-art models only reach a processing rate of less than 10 frames per second [13].

III. Method: LRHPerception

We present a state-of-the-art autonomous perception package designed for monocular image inputs, striking an optimal balance among interpretability, information richness, and computational efficiency, ensuring real-time processing capabilities. Our package's architecture (see Fig.1) encompasses key functionalities of object tracking, trajectory prediction, road segmentation, and depth estimation. Unlike serial connections, we've integrated these modules within our model to facilitate information sharing, reducing redundant input processing.

For common feature recognition, we have chosen transformer backbones, particularly Swin Transformer [16]. Acknowledged for its versatility, Swin Transformer outperforms convolutional backbones, notably in depth estimation [17]. This backbone ingests an RGB image to generate feature-extracted maps Φ_{2^k} with stride size 2^k, where $k \in \{2, 3, 4, 5\}$.

The feature maps, namely $\{\Phi_4, \Phi_8, \Phi_{16}, \Phi_{32}\}$, are routed to the aforementioned modules for subsequent tasks. Specifically, $\{\Phi_8, \Phi_{16}, \Phi_{32}\}$ are relayed to the convolution decoder for simultaneous pixel segmentation and trajectory prediction. Meanwhile, all feature maps contribute to the Depth Former to synthesize a singular depth map layer. The final output of the LRHPerception package is a composite of the original RGB input I, segmentation O, depth map D, and a trajectory prediction overlay $\{Y_{t+1}, ..., Y_{t+\delta}\}$.

Such an integration yields significant computational savings. When detection, tracking, segmentation, and depth estimation are processed individually from the input, computation for three backbones is needed. In contrast, our architectural design leverages a shared backbone and two distinct feature extractions– one from the convolution decoder and the other from the depth estimation module. This strategy lets us complete four tasks with only computation cost for one backbone.

Beyond the computational efficiencies from the integration technique, intrinsic innovations within each functional domain also hold significant importance. These innovations secure efficiency improvements without compromising the efficacy of individual tasks, as detailed in the subsequent sections.

A. Object Tracking

Object tracking stands fundamental to autonomous driving, rooted deeply in object detection paradigms of computer vision [18]. A suite of detection algorithms, namely Faster R-CNN [19], YOLO series [20], and more, are harnessed to bolster tracking performance.

Once detection on a single frame is formulated, the focus of most tracking techniques shifts towards establishing data association across frames. The Kalman filter (KF) [21] is a popular choice to anticipate tracklet locations and attain tracklets matching via location similarity [18]. Additional cues, such as camera movement and low-confidence boxes, have been factored in by various methods to achieve cutting-edge results [22]–[24]. Our module innovates on these cues to produce refined results for tracking robustness and safety:

Our **C-BYTE** (Camera-Calibrated BYTE) approach, de-

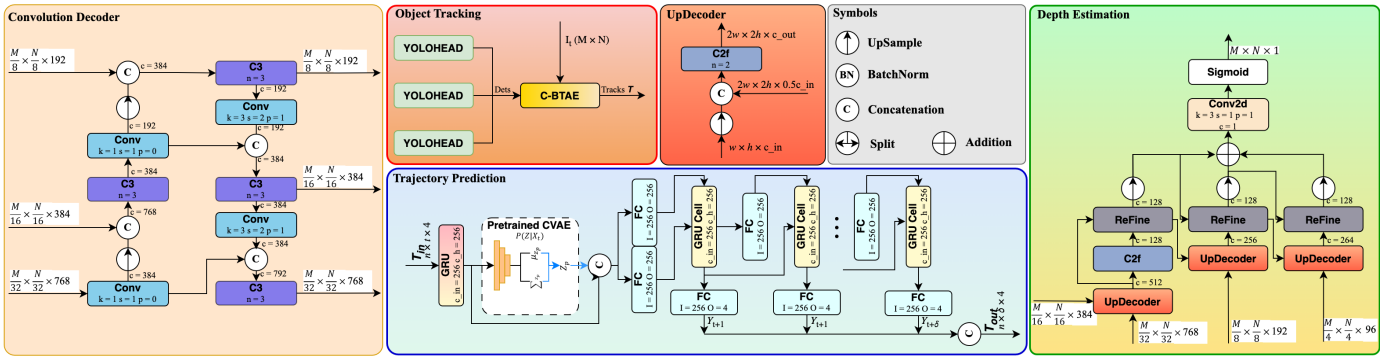

Fig. 2: **Granular Model Structure.1** Design of convolution decoder, object tracking, trajectory prediction, and depth estimation; magnify for details. BTAE mechanism in Algorithm 1. Remaining components are shown in Fig. 3.

picted in Fig.2, is our contribution aimed at formulating tracking trajectories with diminished errors. Distinguished from the original BYTE method [22] that hinges on bounding box correlations, our further strategy incorporates a camera movement correction mechanism between two adjacent frames to refine associations with immediate camera motion. This is particularly pertinent in scenarios of autonomous driving applications where vehicles are constantly moving.

For the processes in focus, consider an input video sequence. After processing to produce a set of bounding boxes from prior modules and Yolohead, the C-BYTE mechanism is activated. First, we perform Lucas-Kanade optical flow between current-time image I_t and previous-time image I_{t-1} to obtain $\mathcal{P}_t$, points with estimated positions in I_t corresponding to the original keypoints in I_{t-1} [25]. The generation of keypoints, $\mathcal{P}_{t-1}$, will be discussed later in this section. Lucas-Kanade method approximates optical flow within a local window around each keypoint, employing spatial and temporal image gradients to solve the flow equation. This least-squares fit computes motion vectors for each point, estimating keypoints' displacement from the previous to the current frame. Then, we calculate the affine matrix $\mathbf{A} \in \mathbb{R}^{2\times3}$ using Random Sample Consensus (RANSAC) with previous and current time keypoints, $\mathcal{P}_{t-1}$ and $\mathcal{P}_t$ [26]. In RANSAC algorithm, a minimal subset of point correspondences is randomly selected to estimate the affine transformation matrix. This matrix is iteratively refined through repeated trials, selecting the model with maximum inliers. The final matrix represents the optimal geometric transformation between frames, robust to outliers.

Now, we separate and apply transformations with the affine matrix following [24], [27]: $\mathbf{A} \in \mathbb{R}^{2\times3} = [R_{2\times2}|O_{2\times1}]$, where R and O represent Cartesian coordinate rotation and displacement matrices respectively. Then we apply these matrices separately to the anticipation of the bounding boxes' positions and velocities $(x, y, w, h; v_x, v_y, v_w, v_h)$ of all current tracks $\mathcal{T}$ in current-time image I_t derived from the Kalman filter: For (x, y), apply displacement transformation: $\begin{bmatrix} x' \\ y' \end{bmatrix} = O + \begin{bmatrix} x \\ y \end{bmatrix}$. For every two variables in these eight parameters, apply rotational transformation similarly to: $\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix}$.

Then, we split the original detection boxes into two lists based on their detection scores: $\mathcal{O}_{\text{high}}$ and $\mathcal{O}_{\text{low}}$. Next, we perform a two-step association following [22]: Primary Association- $\mathcal{O}_{\text{high}}$ are matched with $\mathcal{T}$ using transformed predictions $\mathcal{O}_{pred}$ from the Kalman filter. Secondary Association- We link $\mathcal{O}_{\text{low}}$ with the remaining tracks, denoted as $\mathcal{T}_{\text{remain}}$ using only Kalman filter predictions $\mathcal{O}_{\text{pred-remain}}$. If tracks remain after this step, $\mathcal{T}_{\text{re-remain}}$, persist beyond a specific duration, k, without re-association, they are then discarded.

During both association phases, a cost matrix $\mathbf{C}$ is formed by computing the mIOU cost between every input object detection D_n and saved tracks T^k. Using this matrix, we solve the linear assignment problem to link each matrix row (existing tracks) with at most one unique column (object detections), adhering to the constraint $\min \sum_{i=1}^{|\mathcal{O}|} \sum_{j=1}^{|\mathcal{T}|} c_{ij}$.

Algorithm 1: Pseudo-code of Key Steps in C-BYTE

Input: Detected objects' bounding boxes at current time $\mathcal{O}$, existing tracks $\mathcal{T}$, current frame I_t, previous frame I_{t-1}, previous keypoints $\mathcal{P}_{t-1}$
Output: Updated tracks $\mathcal{T}$
/* Affine Matrix Application */
$\mathcal{P}_t \leftarrow \text{LKOpticalFlow}(I_{t-1}, I_t, \mathcal{P}_{t-1})$
$\mathbf{A} \leftarrow \text{RANSAC}(\mathcal{P}_{t-1}, \mathcal{P}_t)$
$\mathcal{O}_{pred} \leftarrow \text{Transform}(\text{KalmanFilter}(\mathcal{T}), A)$
/* Primary Association */
Split $\mathcal{O}$ to $\mathcal{O}_{\text{high}}, \mathcal{O}_{\text{low}}$ based on detection score
$\mathbf{C} \leftarrow mIOU(\mathcal{O}_{pred}, \mathcal{O}_{high})$
Associate $\mathcal{T}$ and $\mathcal{O}_{high}$ using $\mathbf{C}$
/* Secondary Association */
$\mathcal{O}_{\text{high-remain}} \leftarrow$ remaining object boxes from $\mathcal{O}_{\text{high}}$
$\mathcal{O}_{\text{pred-remain}} \leftarrow$ remaining objects boxes from $\mathcal{O}_{pred}$
$\mathcal{T}_{\text{remain}} \leftarrow$ remaining tracks from $\mathcal{T}$
$\mathcal{T}_{\text{re-remain}} \leftarrow$ remaining tracks from $\mathcal{T}_{\text{remain}}$
$\mathbf{C} \leftarrow mIOU(\mathcal{O}_{pred-remain}, \mathcal{O}_{low})$
Associate $\mathcal{T}_{remain}$ and $\mathcal{O}_{low}$ using $\mathbf{C}$
/* Remove unassociated and initialize new tracks */
$\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{T}_{\text{re-remain}}$
for *each* o *in* $\mathcal{O}_{high\text{-}remain}$ **do**
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{o\}$
/* Update previous frame keypoints */
$\mathcal{P}_{t-1} \leftarrow \emptyset$
$I_l \leftarrow \text{LaplacianOperation}(I_t)$
for *each point* (x, y) *in* I_l **do**
  if $I_l[x, y] > \theta_{th}$ **then**
    $\mathcal{P}_{t-1} \cup \{(x, y)\}$
return $\mathcal{T}$

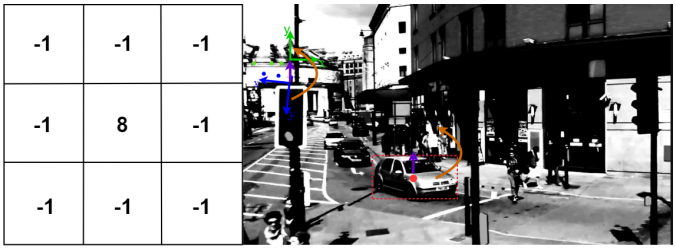

Fig. 3: **Convolution Kernel & Transformation Visualization** On the right, blue and green points represent $\mathcal{P}_{t-1}$ and $\mathcal{P}_t$. Purple and orange arrows denote displacement and rotational transform. Red dashed box is the predicted object position from KF.

$x_{ij} \mid x_{ij} \in \{0, 1\}$, where $|\mathcal{O}|$ and $|\mathcal{T}|$ represents cardinality of $\mathcal{O}$ and $\mathcal{T}$, respectively [28].

The next step is to update current tracks $\mathcal{T}$ using associated tracks after these two associations (new locations from bounding boxes). Leftover detections from $\mathcal{O}_{\text{high}}$ trigger new tracks, with the remaining tracks removed after staying unassociated for a period. Finally, we update the previous frame keypoints $\mathcal{P}_{t-1}$ with those generated from the current frame.

For keypoints generation, we convolve the current image I_t with a discrete approximation of a Laplacian operator, shown in Fig.3. The Laplacian operation, $\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$, highlights regions of rapid intensity change, such as edges and isolated points. Points with a value greater than θ_{th} remaining active, from which we sample a points and save them as $\mathcal{P}_{t-1}$.

A succinct representation of C-BYTE, highlighting critical elements in green, can be found in Algorithm 1.

B. Trajectory Prediction

Trajectory prediction necessitates real-time sensory data, complemented by a system skilled in identifying and tracking traffic elements. Key details such as bounding box dimensions, position, velocity, acceleration, and heading alterations are imperative for this task [18]. Central to this task is the generation of potential future scenarios $Y_\delta | \delta \in [t+1, t+m]$ informed by n past coordinates $X_\eta | \eta \in [t-n, t]$. Here, each X_η and Y_δ represents the locations of the bounding boxes (top-left and bottom-right corner) at time η or δ.

Methodologies in this domain span from Bayesian LSTMs, which utilize observation uncertainty for location predictions [29], to Conv1D frameworks that exploit multi-modal data for predicting pedestrian movements [30]. Goal-driven trajectory prediction has recently gained traction, emphasizing conditional step-by-step forecasting at the cost of computational efficiency [31], [32].

Our design utilizes pre-trained conditional variational autoencoders to provide multi-modal information and encode past trajectories and a refined step-wise goal estimator as a decoder for future trajectories:

Past Trajectory Encoder: When trajectories $\mathcal{T}$ generated by C-BYTE are received, they are converted to a series of past trajectories X_η and undergo an initial encoding process via a Gated Recurrent Unit (GRU). This step refines sequential and contextual details into h_t from X_η and $h_{init} = 0$. It is succeeded by a pre-trained Conditional Variational Autoencoder (CVAE), incorporated from BiTrap [31], which includes a latent prior net $P(Z|X_t)$. Based on the observed trajectories, this latent net predicts an ensemble of mean values μ_{Z_p} and

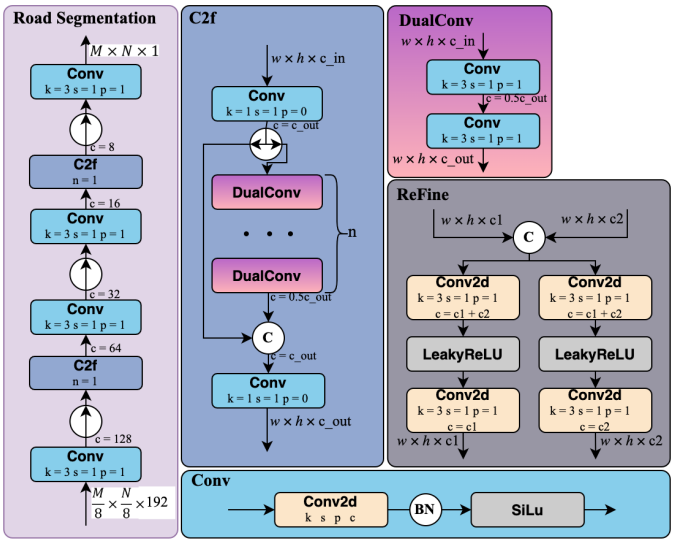

Fig. 4: **Granular Model Structure.2** Design of road segmentation block, along with other components.

covariance Σ_{Z_p} of all conceivable future positions within the time frame $\delta \in [t+1, t+\delta]$. Then, Z, an array of mean and covariance samples, is drawn from $N(\mu_{Z_p}, \Sigma_{Z_p})$. We then concatenate these latent variables to h_t, $(h_t \oplus Z)$, to inform the decoder of possible future positions in which objects can reside. Such architecture design facilitates faster extraction of salient and latent information by removing the need to rely on a succession of Recurrent Neural Networks [32].

Future Trajectory Decoder: Receiving concatenated output from the encoder, the decoder sequentially projects objects' future trajectories. Recognizing a single GRU block, generating the entire sequence of future time-step hidden states $h_{t+1}, ..., h_{t+m}$ from h_t, can only encapsulate data up to time t, we employ a sequence of discrete GRU Cells. Each individually yields h_{t_i+1} from h_{t_i} containing data up to time t_i, where $t_i \in [t, t+m-1]$. After each cell, we use a fully-connected layer to derive the final numerical location output $Y_\delta | \delta \in [t+1, t+m]$.

During implementation, two problems arise: 1) the encoder output $(h_t \oplus Z)$, a concatenation of hidden states and latent variables, may deviate from the domain of the GRU cell's hidden state input h. Thus, to reduce this domain gap, we introduce another fully-connected (dense) layer to convert $(h_t \oplus Z)$ into h_{tz}. 2) We recognize an absence in input vector Y_{t_i} for the GRU cells at future times t_i. And these input vectors differ from the decoded location output Y_{t_i} as it must be interpretable to the GRU cell. Therefore, we append dense layers to transmute the output of the previous GRU cell, h_{t_i-1}, into the input vector Y_{t_i}.

The architecture preserves additional information, ensuring smooth data flow and enabling efficient one-directional predictions that capture comprehensive data dependencies.

C. Road Segmentation

Segmentation is a technique to divide images into regions, crucial for pinpointing and isolating objects of interest, including semantic and instance segmentation [33]. Recent methodologies propose universal image segmentation, aiming to classify a wide range of Objects [34]–[36]. However, our module adopts a more focused approach, specifically designed

for autonomous driving by confining the segmentation scope to drivable surfaces. The intent is to reduce the computational load necessary for high-resolution segmentation tasks.

Segmentation block's architecture, illustrated in Fig.4, is minimalist but can achieve noteworthy results. Given a concentrated focus on one class, we adopt the U-Net framework, where $O = D(E(I))$ [37]. Determining our $E(I)$, we recognize that the stride-8 feature map, Φ_8, from our convolutional decoder encompasses decoded data from strides 16 and 32. Decoding this feature map removes any additional processing needed for Φ_{16} and Φ_{32}, enhancing processing speed. Thus, the mechanism of our block becomes $O = D(\Phi_8)$.

Decoder's design aligns with a convolutional decoding blueprint, melding CBS (Conv2D-BatchNorm-SiLu) with C2f that consists of regular convolutions, DualConv, and skip connections adapted from YOLO [20] and shown in Fig.4. Here, we replace the traditional Bottleneck block with a dual-CBS setup, further optimizing computational efficiency. By integrating these with upsampling layers, the module swiftly decodes Φ_8 into a tensor $S_{h,w}$, congruent in dimensions to input I, representing the pixel-level road classification. This specialized approach enables our model to clearly discern drivable surfaces with faster speed than broader models.

D. Depth Estimation

The pursuit of depth estimation from single images, crucial for robotic navigation and autonomous driving [17], has evolved significant, especially with the rise of deep learning. Depth learning strategies fall into three primary categorizations based on their constraints. The first strategy leverages ordinal relation constraints, employing listwise ranking mechanisms [38]. The second approach emphasizes surface normal constraints, refining depth prediction borders and recognizing long-range relationships [39]. Lastly, the heuristic refinement category focuses on enhancing post-prediction depth [40].

Advanced methodologies often blend elements from these classifications, using variational constraints followed by refinements for precision [41]. Our module, inspired by this approach, aims to better balance computational expense with accuracy. Conceptually, depth formation mirrors single-class segmentation; the goal is to associate each pixel in an input image with a specific depth value, articulated as $\sum_{i=1}^{M}\sum_{j=1}^{N} P(i,j,I) = D(i,j)$. Here, M and N represents width and height of the input image I. The emphasis is to devise a competent decoder, exemplified by the function P. In alignment with the widely adopted Encoder-Coarse-Refine methodology in monocular depth estimation [41], we delineate the following modules for our depth estimation process:

Coarse Depth Former: This module operates predominantly on condensed feature maps, establishing global depth references within images for subsequent refinements. Hence, We selected the richest layers from backbone outputs Φ_{16} and Φ_{32} for decoding purposes. To enhance processing rates, we engineered a straightforward UpDecoder, employing C2f as the primary decoding conduit for the input feature maps. An auxiliary C2f layer then formulates the preliminary depth map $D(i,j)$ for images, where $i \in [0, M/8], j \in [0, N/8]$. Such a simple configuration, therefore, offers a reduction in processing duration relative to other methods [41] while maintaining comparable efficacy.

Refine Depth Former: This module's role is refining the initial depth input to produce an exact depth layer commensurate with input image I. To accomplish this, we once again employ U-Net, but with a tailored configuration. Instead of directly merging the upscaled depth map with the backbone feature maps $(D(i,j) \oplus \Phi_b)$ and using convolutions to form $D(i^*, j^*)$, we integrated output from a secondary flow in the refinement block. This flow has the same structure as the main flow that produces the output depth map, whose outputs are subsequently fused with the backbone feature map by the UpDecoder for further refinement. Expressed succinctly, $D(i^*, j^*) = R(D(i,j) \oplus \Phi_{UD})$, where $\Phi_{UD} = UD(\Phi_R \oplus \Phi_b), i^* \in [0, 2i], j^* \in [0, 2j]$. Utilizing multiple such refinement layers, the final depth map $D(i_f, j_f)$ is a blend of several refined outputs $D(i^*, j^*)$, capturing details across varied scales while retaining a cohesive module structure.

E. Training and Loss Function

One challenge of training such a multi-task model is the lack of a single, comprehensive dataset that covers every module. To address this, we adopt a cross-dataset training approach. Rather than limiting our model to a singular dataset, we train individual modules on multiple datasets, each known for its strengths in specific domains.

For instance, the Kitti dataset [42] specializes in monocular depth estimation and object detection. Similarly, the Cityscape dataset [43] is used for our road segmentation module. As these modules are trained, they collaboratively refine the learnable parameters in the backbone and convolution decoder. This means that Swin transformer backbone becomes a task-agnostic module tuned on both Kitti and Cityscape datasets. On the other hand, our trajectory prediction module learns from the JAAD [44] and PIE [45] datasets, which feature marked pedestrian and vehicle trajectories (Cartesian coordinates) from monocular camera videos and thus do not require the involvement of previous modules.

Our approach to integrating domain-specific losses is summarized in the equation: $L = \lambda_{det}L_{det} + \lambda_{seg}L_{seg} + \lambda_{depth}L_{depth} + \lambda_{traj}L_{traj}$, with all parameters subject to optimization. For the weightings, we assign a value of 5 to λ_{seg} from empirical findings, while keeping λ_{det}, λ_{depth}, and λ_{traj} at a balanced value of 1.

IV. Experiments & Results

Our experimentation framework comprises two primary components: modular analysis to underscore the robustness of individual innovation on quantitatively testable datasets, and comprehensive assessment to exemplify our fusion techniques alongside information-rich representation of the environment. Despite parallel training, we exclusively utilize a single RTX 3090 GPU for all tests, with metrics lower the better unless stated otherwise. Bold values represent our methods, while underlined ones signify the top-performing results.

A. Modular Results

For modular analysis, a complete cross-dataset trained package is tested on all tasks except object tracking. Comparison models are singularly trained for their specific tasks. As our

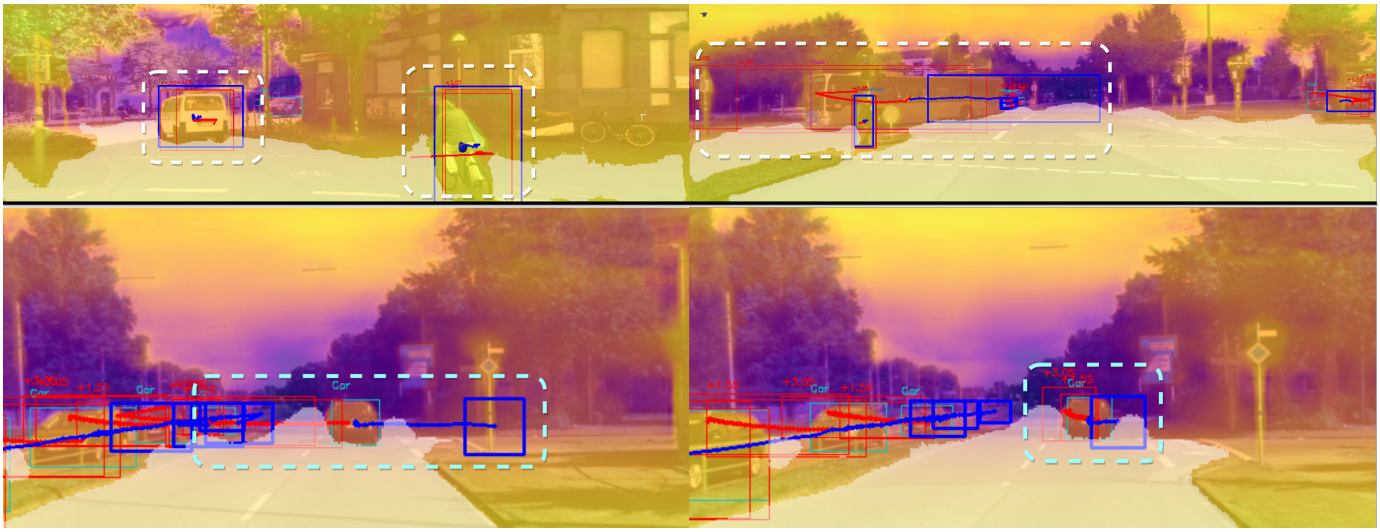

Fig. 5: **Results Visualization** The quartet of images depicts the output of LRHPerception, with past trajectories delineated in blue and future trajectory predictions in red. The **upper** pair of images exemplifies two **Success Cases**, while the **lower** duo present one **Failure Case**.

Methods	Detect	MOTA↑	IDF1↑	IDP↑	Time(ms)
C-BYTE(Ours)	**YoloX**	**76.9%**	**81.2%**	**85.9%**	**31.0**
OC(2023) [46]	YoloX	74.1%	77.8%	87.2%	28.3
Byte(2022) [22]	YoloX	76.6%	79.3%	84.0%	27.1
BoT(2022) [24]	YoloX	76.8%	81.0%	85.7%	48.2

TABLE I: **Object Tracking.** Our model manifests a significant improvement over SOTA across all efficacy and efficiency metrics on MOT datasets, corresponding harmoniously with our structure- fundamentally predicated on achieving refined outputs for robustness. Hyperparameter θ_{th} and a are empirically determined as 0.9 and 210.

Methods **Dataset: JAAD**	MSE (0.5/ 1.0/ 1.5s)	C_{MSE} (1.5s)	CF_{MSE} (1.5s)	FPS ↑ (8 / 12 / 24)
LRHP (Ours)	**43/ 113/ 283**	**239**	**662**	**111/ 104/ 92.6**
SGNet (2021)	82/ 328/ 1049	995	4076	2.8/ 2.8/ 2.7
Bitrap (2020)	93/ 378/ 1206	1105	4565	99.3/ 105/ 97.1
PIE_traj (2019)	110/ 399/ 1280	1183	4780	-/ -/ -
B-LSTM (2017)	159/ 539/ 1535	1447	5615	-/ -/ -
Dataset: PIE				
LRHP (Ours)	**19/ 44/ 104**	**81**	**233**	**111/ 104/ 92.6**
SGNet (2021)	34/ 133/ 442	413	1761	2.8/ 2.8/ 2.7
Bitrap (2020)	41/ 161/ 511	481	1949	99.3/ 105/ 97.1
PIE_traj (2019)	58/ 200/ 636	596	2477	-/ -/ -
B-LSTM (2017)	159/ 539/ 1535	1447	5615	-/ -/ -

TABLE II: **Trajectory Prediction.** Our model exhibits significant enhancements in all aspects of efficacy and efficiency relative to SOTA methods, particularly pronounced as prediction time expands.

C-BYTE for object tracking lacks learnable parameters, we employ ByteTrack's framework as the benchmarking standard.

1) Object Tracking

We evaluate C-BYTE within ByteTrack's framework to use the same backbone of YoloX-x, following the "private detection" protocol across MOT17 (multiple-object-tracking) datasets [47]. We employ the conventional metrics of MOTA (Multiple Object Tracking Accuracy), IDF (ID F1 Score), and IDP (ID Precision), to assess various facets of tracking: MOTA is based on FP (false positive), FN (false negative) and IDs, while IDF and IDP evaluate association performance by penalizing inaccurate tracks.

The resultant data in Table.I exhibits a noticeable advancement over Byte and other SOTA methods, substantiating that camera motion correction refined the Kalman Filter's predictions by removing the nonlinear disturbance for which linear models of KF cannot account. With a negligible delay of less than 4 milliseconds compared to Byte, our method demonstrates superior tracking results across metrics. The sturdiness of this module is further validated in the visualizations of joint tests, where we observe minimal to nonexistent instances of failure attributable to the trajectory former. This empirical evidence corroborates the efficacy and reliability of our model.

Methods	mIOU ↑	FPS ↑	Remarks
LRHP (Ours)*	**88.9**	**55.0**	* speed on dual
Road Segmentation (Yolo Backbone)*	**88.4**	**96.4**	tasks of detecti-
InternImage (2023) [36]	86.1	79.5	on and segmen-
MSeg (2020) [34]	77.6	65.7	tation
UJS-base (2021) [35]	80.5	-	
UJS-refined (2021) [35]	88.3	-	

TABLE III: **Road Segmentation.** Our model exhibits an appreciable acceleration and precision enhancement when operating an additional detection task with the YOLO backbone- a testament to the robustness and effectiveness of our architectural design.

Methods	Backbone	RMS	δ_1 ↑	δ_2 ↑	FPS ↑
LRHP (Ours)	**Swin-m**	**0.229**	**0.966**	**0.996**	**42.0**
Depth Estimate	**Swin-L**	**0.216**	**0.975**	**0.997**	**13.3**
VA-Depth (2023)	Swin-L	0.209	0.977	0.997	6.2
AdaBins (2021)	EffNet&Vit	0.236	0.964	0.995	1.7
BTS (2019)	DenseNet	0.280	0.955	0.993	20.1
ASTrans (2021)	Vit-B	0.269	0.963	0.995	-
DORN (2018)	ResNet	0.273	0.932	0.984	-

TABLE IV: **Depth Estimation.** Our model showcases a significant acceleration in processing speed relative to SOTA, all the while maintaining comparable accuracy levels, thus underscoring the remarkable efficiency of our design.

2) Trajectory Prediction

Our prediction former is assessed on JAAD and PIE datasets [44], [45], which feature ego-centric videos annotated at 30Hz. Following established benchmarks [45], we utilize a 15-frame observational duration and a 45-frame prediction horizon for the evaluation, where the ground-truth observation is given. We additionally introduce a pragmatic speed assessment wherein we aggregate tracks in batches of 8, 12, and 24, indicative of the object count within a single image, and ascertain how many batches can be processed per second.

The outcomes in Table.II evince a distinct augmentation in both speed and accuracy across both datasets, a discrepancy that broadens as the prediction timeline extends into the future. In quantitative terms, our model presents an impressive **40-fold** increase in processing speed compared to the alternate highest-accuracy method [32] and a **4-fold** boost in accuracy over the quickest model [31]. Such robust performance attests to the efficacy of an encoder capturing explicit and latent dependency and a decoder featuring swift unidirectional prediction.

*3) **Road Segmentation***

Our segmentation former is evaluated on Cityscape dataset [43]. The dataset encompasses pixel segmentation across all classes present, with our module specifically targeting drivable surfaces. Tests are conducted on the validation set with models presenting mIOU (mean intersection over union) across the entire set. During speed evaluation, we execute both detection and segmentation as our module is integrated within a convolution decoder for object detection. We also implement a standalone segmentation module using the YOLO backbone to illustrate the simplicity of the segmentation module itself.

Remarkably, our module surpasses the performance of prevailing universal modules in Table.III, underscoring the potency of simplicity in certain domains. Although our final choice of the Swin backbone, geared towards combined modules, increases segmentation accuracy but reduces processing speed, this module still excels in executing its designated task in subsequent joint tests, contributing to the overall goal of real-time monocular perception.

*4) **Depth Estimation***

Our depth estimator is assessed on the KITTI dataset [42] adhering to the splits delineated in [41], where depth maps are annotated with a range of 0 to 80 meters. A selection of quintessential metrics, including RMS (root mean square error) for overall error estimation and δ_1 and δ_2 for precision within specific tolerances, are employed. To underscore the efficacy of our module design, we also implement a standalone depth estimator with the Swin-L backbone, neutralizing the computational variances on backbones to evaluate our depth estimation decoder design.

The yielded results in Table.IV underscore a noteworthy acceleration in frames processed per second, whilst maintaining a high degree of accuracy. Concretely, our design manifests a **577%** uplift in processing speed over the best-alternative [41]. Should the same backbone be used, our decoder design alone offers **115%** improvement in speed with comparable accuracy. These values validate the design of a simplified coarse-refine layout using modified C2f layers. The success of this module affords our suite the ability to produce accurate vital depth information without sacrificing real-time capabilities.

*5) **Ablation Study***

We studied the choice of hyperparameter n in our introduction of the C2f module into road segmentation and depth estimation, presented in Table.V. With this study, we incorporated two DualConv modules in the C2F to balance the efficacy and computational need.

B. Joint Results

We carry out comprehensive assessments on KITTI Dataset [42], featuring videos captured at a rate of 10 frames per second. This selection of the dataset serves to highlight the practical efficacy of our model under real-world scenarios, bolstering its applicability and value for future research.

Number of DualConv	Road-Seg (mIOU)↑	Depth Estimation (RMS)	Time Difference ($T_n - T_{n=1}$)(ms)
$n = 1$	88.5	0.223	0
$n = 3$	88.5	0.219	0.268
$n = 2$	**88.9**	**0.220**	**0.049**

TABLE V: **Ablation study.** We chose $n = 2$ with an mIOU of 88.9 for road segmentation and an RMS of 0.220 for depth estimation, alongside a minimal time increase of 0.049 ms over the baseline.

Methods	Category	FPS ↑
LRHP(Ours)	**Monocular**	**28.8**
LRHP in series	Monocular	16.3
SOTA in series	Monocular	1.8
Uni-AD (2023) [13]	Multi-Cam Map	2.1
BEVerse-Tiny (2022) [14]	Multi-Cam Map	4.4
DETR3D (2021) [5]	Multi-Cam Map	2.0

TABLE VI: **Entire Model.** Our model witnesses an improvement of more than an order of magnitude over existing local mapping methods, a testament to the exceptional efficiency of our model. Note that Uni-AD's planning module was removed for a fair comparison.

Quantitative. In our empirical comparison, we juxtapose the computational demands of our method against multi-camera map techniques and monocular methods in Table.VI. The latter are constructed from the current SOTA solutions in each domain of tracking, trajectory prediction, road segmentation, and depth estimation. Remarkably, LRHPerception facilitates a real-time processing rate of **29** FPS, constituting a substantial **555%** acceleration over the fastest mapping technique. Upon scrutinizing the contributions to this efficiency, our module enhancements account for an 806% acceleration relative to sequentially-connected SOTA methods, with our integration technique further doubling the speed-up to **1500%**.

Qualitative. Given that our perception package represents an unparalleled fusion of functionalities, we resort to visualization for qualitative assessment. Fig.3 encapsulates successful instances, embodied in white boxes, including a right-turning van and a forward-moving bicycle along a road and a left-turning bus and a stationary pedestrian in an intersection. Recognizing the pedagogical value of shortcomings, we also feature failure cases. One typical scenario illustrates a right-turning car mispredicted to continue leftward, with the correct forward trajectory identified half a second later. Furthermore, the segmentation module overlooks a potential route to the right of the intersection. These areas of discrepancy delineate the LRHPerception for future enhancements.

V. Conclusion

In this work, we unveil LRHPerception, a monocular perception package that achieves a balanced information richness and computational load. It efficiently and seamlessly blends road identification, object surveillance, trajectory prediction, and distance approximation, aligning ego-planner with human perception, achieving real-time functionality while offering human-understandable interpretation to the surrounding environment. Serving as a humble groundwork for exploration, LRHPerception constitutes an efficient toolkit, laying work for future ingenuity in the safe comprehensible autonomous driving domain.

References

[1] Z. Wu, F. Qu, L. Yang, and J. Gong, "Human-like decision making for autonomous vehicles at the intersection using inverse reinforcement learning," *Sensors*, vol. 22, no. 12, p. 4500, 2022.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] Y. Hou, S. Hornauer, and K. Zipser, "Fast recurrent fully convolutional networks for direct perception in autonomous driving," *arXiv preprint arXiv:1711.06459*, 2017.

[4] L. Heng, B. Choi, Z. Cui, M. Geppert, S. Hu, B. Kuan, P. Liu, R. Nguyen, Y. C. Yeo, A. Geiger, *et al.*, "Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4695–4702.

[5] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," in *Conference on Robot Learning*. PMLR, 2022, pp. 180–191.

[6] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 222–17 231.

[7] J. Norden, M. O'Kelly, and A. Sinha, "Efficient black-box assessment of autonomous vehicle safety," *arXiv preprint arXiv:1912.03618*, 2019.

[8] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1364–1384, 2020.

[9] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang Jr, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, 2017.

[10] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in Neural Information Processing systems*, 1988.

[11] A. Amini, A. Soleimany, S. Karaman, and D. Rus, "Spatial uncertainty sampling for end-to-end control," *arXiv preprint arXiv:1805.04829*, 2018.

[12] S.-G. Jeong, J. Kim, S. Kim, and J. Min, "End-to-end learning of image based lane-change decision," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1602–1607.

[13] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[14] Y. Zhang, Z. Zhu, W. Zheng, J. Huang, G. Huang, J. Zhou, and J. Lu, "Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving," *arXiv preprint arXiv:2205.09743*, 2022.

[15] J. Gu, C. Hu, T. Zhang, X. Chen, Y. Wang, Y. Wang, and H. Zhao, "Vip3d: End-to-end visual trajectory prediction via 3d agent queries," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5496–5506.

[16] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[17] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[18] F. Leon and M. Gavrilescu, "A review of tracking and trajectory prediction methods for autonomous driving," *Mathematics*, vol. 9, no. 6, p. 660, 2021.

[19] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*.

[20] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[21] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems [j]," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[22] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *European Conference on Computer Vision*, 2022.

[23] X. Hou, Y. Wang, and L.-P. Chau, "Vehicle tracking using deep sort with low confidence track filtering," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2019, pp. 1–6.

[24] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[25] N. Sharmin and R. Brad, "Optimal filter estimation for lucas-kanade optical flow," *Sensors*, vol. 12, no. 9, pp. 12 694–12 709, 2012.

[26] K. G. Derpanis, "Overview of the ransac algorithm," *Image Rochester NY*, vol. 4, no. 1, pp. 2–3, 2010.

[27] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. London, England: Springer, 2008.

[28] S. Martello and P. Toth, "Linear assignment problems," in *North-Holland Mathematics Studies*. Elsevier, 1987, vol. 132, pp. 259–282.

[29] A. Bhattacharyya, M. Fritz, and B. Schiele, "Long-term on-board prediction of people in traffic scenes under uncertainty," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4194–4202.

[30] T. Yagi, K. Mangalam, R. Yonetani, and Y. Sato, "Future person localization in first-person videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7593–7602.

[31] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1463–1470, 2021.

[32] C. Wang, Y. Wang, M. Xu, and D. J. Crandall, "Stepwise goal-driven networks for trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2716–2723, 2022.

[33] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021.

[34] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, "Mseg: A composite dataset for multi-domain semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2879–2888.

[35] L. D. Yingfeng Cai and Z. L. Hai Wang, "Multi-target pan-class intrinsic relevance driven model for improving semantic segmentation in autonomous driving," in *IEEE Transactions on Image Processing (TIP)*, November 2021.

[36] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, *et al.*, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[37] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[38] J. Lienen, E. Hullermeier, R. Ewerth, and N. Nommensen, "Monocular depth estimation via listwise ranking using the plackett-luce model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 595–14 604.

[39] W. Yin, Y. Liu, C. Shen, and Y. Yan, "Enforcing geometric constraints of virtual normal for depth prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5684–5693.

[40] W. Yuan, X. Gu, Z. Dai, S. Zhu, and P. Tan, "Neural window fully-connected crfs for monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3916–3925.

[41] C. Liu, S. Kumar, S. Gu, R. Timofte, and L. Van Gool, "Va-depthnet: A variational approach to single image depth prediction," *arXiv preprint arXiv:2302.06556*, 2023.

[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[43] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[44] I. Kotseruba, A. Rasouli, and J. K. Tsotsos, "Joint attention in autonomous driving (jaad)," *arXiv preprint arXiv:1609.04741*, 2016.

[45] A. Rasouli, I. Kotseruba, T. Kunic, and J. K. Tsotsos, "Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6262–6271.

[46] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9686–9696.

[47] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A

benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.