LaTeX Author Guidelines for ICCV Proceedings

Anonymous ICCV submission

Paper ID *****

Abstract

This paper explores the application of digital twin technology in autonomous driving, introducing the concept of Urban Vehicle Digital Twin (UVDT). By studying vehicle detection, target tracking, re-identification, and trajectory reconstruction, the traffic flow is replicated. The use of digital twin technology combined with sensor data improves the accuracy and robustness of vehicle detection. The virtual environment optimizes multi-target tracking performance, addressing challenges such as occlusion and changes in viewpoint. In the area of re-identification, the generation of rich training data enhances cross-scene vehicle re-identification capabilities. The vehicle control method based on digital twin provides high-precision path planning and control for autonomous driving systems, ensuring accurate navigation in complex traffic scenarios. Experimental results show that UVDT effectively optimizes various modules of the autonomous driving system, advancing intelligent transportation systems and offering important support for the development of future smart cities.

1. Introduction

With the rapid development of intelligent transportation, autonomous driving, and urban management, digital twin technology, as an important innovative tool, has gradually shown tremendous potential in various fields[5]. A digital twin refers to the creation of a digital model corresponding to the real world through the real-time synchronization of data collected from the physical world and virtual models[17]. This technology can simulate, analyze, and optimize the real world in a virtual environment, thus providing support for decision-making[1]. In the field of autonomous driving, digital twin technology can accurately replicate factors such as traffic flow, road structure, and pedestrian behavior, providing a precise testing and training environment for the perception, planning, and control of autonomous driving systems[15][10].

The focus of this study is to apply digital twin technology to autonomous driving systems, enhancing the interpretability and robustness of the system by replicating real traffic flow scenarios[24]. By constructing high-precision digital twin models and combining reinforcement learning with simulation platforms, we are able to simulate complex driving scenarios and train and optimize autonomous driving algorithms[9]. At the same time, digital twin technology can provide smarter decision support for traffic management, driving the development of smart city construction[6].

We primarily conduct research on digital twins from four aspects: vehicle detection, object tracking, re-identification, and trajectory reproduction. In vehicle detection, we enhance detection accuracy by combining digital twin technology with sensor data. In object tracking, we use virtual environments to address challenges posed by occlusions and viewpoint changes, optimizing multi-object tracking performance. In re-identification tasks, we generate rich training data through digital twin models, improving vehicle re-identification across different scenes. In trajectory reproduction, digital twins provide high-precision path planning and control for autonomous driving systems, simulating real traffic flow scenarios to ensure precise vehicle movement. Through research in these four areas, we drive the optimization of autonomous driving technologies and the development of intelligent transportation systems.

Vehicle detection is a fundamental task in autonomous driving systems, typically relying on data from various sensors such as LiDAR, cameras, and millimeter-wave radar for object recognition and localization. In recent years, the application of deep learning technologies has significantly improved the accuracy and robustness of vehicle detection. Through digital twin technology, we can create precise virtual environments to provide abundant training data for autonomous driving systems, enhancing detection performance. Digital twins not only improve detection effectiveness in complex and dynamic environments but also optimize the fusion of data from different sensors[3].

Object tracking is a key technology in autonomous driving, particularly in multi-object tracking, where the task becomes especially complex due to target occlusions, changes in viewpoint, and misalignment between multiple cameras.

Digital twin technology, by simulating the behavior of targets in different scenarios, can provide high-quality training data for multi-object tracking while also simulating occlusions and dynamic changes. Combined with technologies such as reinforcement learning, digital twins help enhance the accuracy and real-time performance of object tracking[26].

Vehicle re-identification (Re-ID) refers to recognizing the same vehicle at different times and locations, especially across multiple viewpoints and different cameras. Digital twin technology provides an accurate virtual environment that generates a large amount of vehicle data with varying perspectives and environmental changes for training re-identification algorithms. In this way, digital twins not only enhance cross-scenario re-identification capabilities but also improve the recognition accuracy of similar vehicles in complex environments[19].

Trajectory replication refers to simulating the movement trajectory of a vehicle, especially in complex urban road environments. Through digital twin technology, we can accurately simulate traffic flow, road structures, and driving routes, providing a platform for testing and validation for autonomous driving systems. Trajectory replication not only helps train autonomous driving control algorithms but also optimizes path planning, ensuring precise navigation of the vehicle in the real world[8].

2. Related Work

We have summarized research work on different aspects of UVDT.

Vehicle Detection. UVDT mainly uses LiDAR and cameras for vehicle object detection. Several recent studies have explored the fusion of LiDAR and camera data for vehicle detection,demonstrating significant improvements in both accuracy and robustness. For instance, MVDNet (2018) utilizes a multi-view fusion approach, where LiDAR point clouds are projected to different views (such as bird's-eye view and front view) and matched with camera images, combining depth information from LiDAR and texture information from the camera. This method improves detection performance, especially in complex environments. Building on this, PointPillars introduces a fast encoder for LiDAR data, using a grid-based encoding approach that could potentially be integrated with camera data in future fusion methods to enhance detection speed and efficiency[13]. Another study combines LiDAR and camera data by feeding them into separate convolutional neural networks, where the features extracted from both modalities are fused to perform object detection[27]. This approach has shown significant improvements in detection performance, particularly in dynamic environments with occlusions or sparse LiDAR data. Lastly, ST-MVDNet introduces a self-training framework using a teacher-student mutual learning mechanism, where the teacher network is trained on the fused LiDAR and camera data, while the student network is exposed to strong data augmentation simulating missing sensor modalities[14].

This approach enhances the model's robustness against sensor failures by ensuring consistency between the teacher and student models, allowing the system to better handle missing or noisy data during inference. These methods collectively highlight the importance of sensor fusion and advanced learning techniques in achieving robust and accurate vehicle detection, even in challenging conditions. We use the PointPillars deep learning method which has shown good performance.

Multi camera multi-target tracking. Most existing algorithms, with a few exceptions, can be seen as special cases of the multi-modal fusion problem. These methods organize the input data using a graph structure, where edges represent relationships between modalities, and nodes represent different targets or states. Algorithms that can be solved in polynomial time typically handle specific modalities or time-continuous edges, with some also utilizing maximum flow or matching algorithms. Methods that leverage global information (beyond just time continuity or modality constraints) can significantly improve performance, but they are usually NP-hard due to the involvement of combinatorial optimization. In some cases, marginal terms or local constraints are added to ensure completeness. To enhance model expressiveness, some studies have employed higher-order relations, although the gains diminish significantly as complexity increases. Joint optimization and iterative optimization strategies have also been widely used to improve performance. We use Joint Integrated Probabilistic Data Association (JIPDA), which combines data from multiple sensors and optimizes probabilistic associations to effectively handle data uncertainty and missing information in target tracking, improving tracking accuracy and robustness in complex environments.

Vehicle re identification. He and his team proposed a part-based regularization method, which enhances the accuracy of vehicle re-identification by detecting local parts of vehicles, such as lights and wheels[7]. Zheng and his colleagues introduced a large-scale vehicle re-identification dataset named VehicleNet and proposed a multi-task learning framework that combines vehicle re-identification and vehicle attribute recognition tasks, improving the model's generalization capability[28]. P and the research team proposed a dual-path model that integrates global and local features and incorporates an adaptive attention mechanism to enhance the accuracy of vehicle re-identification[12].

The most advanced technologies in the field of vehicle re-identification currently include the combination of deep convolutional neural networks (CNN) for feature extraction and metric learning[11], particularly with the integration

of cross-view and cross-domain learning techniques, the use of Generative Adversarial Networks (GAN) for image enhancement[4], and the fusion of multi-sensor data (such as cameras, radar, and LiDAR) to improve the model's robustness and accuracy in complex environments[16].

We designed a ResNet-50 network for vehicle re-identification by drawing inspiration from Re-ID. The network is capable of performing this task effectively.

twin. After obtaining all the data, we will proceed with vehicle control. Kaleb Ben Naveed et al. proposed a hierarchical reinforcement learning-based method for autonomous driving trajectory planning and control. By training a reinforcement learning agent in a simulation environment, this method optimizes the vehicle's control strategy according to environmental changes, achieving efficient trajectory planning and dynamic path adjustment.[21] R. Barea et al. proposed a deep reinforcement learning (DRL)-based control method, where an agent is trained in the CARLA simulation platform to autonomously learn and optimize the vehicle's control strategies (such as throttle, brake, and steering) to ensure safe and smooth driving.[20] We use a trajectory smoothing algorithm to control the vehicle's trajectory, enabling the vehicle to more accurately follow the planned path, especially in situations with many road curves or irregular road surfaces. Compared to precise real-time optimization control, the trajectory smoothing algorithm typically involves fewer real-time computations, reducing reliance on computational resources while maintaining effectiveness and improving real-time performance.

3. Method

3.1. Problem definition

In UVDT, the input for single-intersection vehicle detection and tracking consists of point cloud data and images, where the point cloud data is the input from the radar and the images are the input from the camera. The input for multi-intersection object detection is the same as that for a single intersection, but the input for its object tracking includes trajectories and the corresponding vehicle appearance features. The input for the re-identification module is a 1×2048 vehicle feature vector. The input for the twin part includes trajectory data (comprising the desired vehicle trajectory and the current vehicle state), environmental information (such as road conditions and obstacle information), and the system's control strategy.

PointPillars. We use the PointPillars network to process the input for vehicle detection and generate the output. A PointPillars network requires two inputs: pillar indices as a P-by-2 and pillar features as a P-by-N-by-K matrix. P is the number of pillars in the network, N is the number of points per pillar, and K is the feature dimension. The network begins with a feature encoder, which is a simplified PointNet. It contains a series of convolution, batch-norm, and relu layers followed by a max pooling layer. A scatter layer at the end maps the extracted features into a 2-D space using the pillar indices. Next, the network has a 2-D CNN backbone that consists of encoder-decoder blocks. Each encoder block consists of convolution, batch-norm, and relu layers to extract features at different spatial resolutions. Each decoder block consists of transpose convolution, batch-norm, and relu layers. The network then concatenates output features at the end of each decoder block, and passes these features through six detection heads with convolutional and sigmoid layers to predict occupancy, location, size, angle, heading, and class.

ReID.

3.2. Single intersection multi-target tracking

TrackerJPDA.

Single Detection Class Probability Update. First, consider the class information association between one detection and one track. Assume the confusion matrix of the detection is $C = [c_{ij}]$,where c_{ij} denotes the likelihood that the classifier outputs the classification as j if the truth class of the target is i. Here,$i, j = 1, \ldots, N$, and N is the total number of possible classes. At time $k-1$, the probability distribution of a track is given as

$$\mu(k-1) = \begin{bmatrix} \mu_1(k-1) \\ \vdots \\ \mu_N(k-1) \end{bmatrix} \tag{1}$$

where μ_i is the probability that the classification of the track is i. If the tracker associates the detection to the track at time k, then the updated value of μ_i using Bayes' theorem is

$$\mu_i(k) = \frac{c_{ij}\mu_i(k-1)}{\sum_{l=1}^{N} c_{lj}\mu_I(k-1)} \tag{2}$$

Write this equation in a vector form for all possible classifications as

$$\mu(k) = \frac{c_j \otimes \mu(k-1)}{c_j^T \mu(k-1)} \tag{3}$$

where c_j is the j-th column of the confusion matrix and $\otimes$ represents element-wise multiplication. This equation represents the updated class probability of a track if the track is associated with the detection of classification j.

Mixed Association Likelihood in Cluster. The tracker performs gating and clustering by using only the kinematic information between detections and tracks. After that, in each cluster, the tracker calculates the mixed likelihood of association between a detection m and a track t as:

$$\Lambda(m,t) = \Lambda_k^{1-\alpha}(m,t)\Lambda_c^{\alpha}(m,t) \tag{4}$$

where α represents Weight factor of class likelihood;Λ_k represents Likelihood of assigning a detection to a track based on the kinematic states;Λ_c represents Likelihood of assigning a classified detection to a track based on the class information. In the equation, Λ_c takes one of these three forms based on the value of m and t.

• $m > 0$ and $t > 0$ — A hypothesis that the measurement is associated with a track in the tracker. In this case,

$$\Lambda_k(m,t) = c_j^T \mu(k-1) \tag{5}$$

where c_j is the j-th column in the confusion matrix that detection m corresponds to, and $\mu(k-1)$ is the class probability vector of the track in the previous step.

• $m > 0$ and $t = 0$ — A hypothesis that the measurement is not associated with any track in the tracker. In this case,

$$\Lambda_k(m,t) = c_j^T \mu^0 \tag{6}$$

where c_j is the j-th column in the confusion matrix that detection m corresponds to, and μ^0 is the a priori class distribution vector of tracks.

• $m = 0$ and $t > 0$ — A hypothesis that the track is not associated with any measurement in the tracker. In this case,

$$\Lambda_k(m,t) = 1 \tag{7}$$

Using the mixed likelihoods of association between detections and tracks in a cluster, the tracker generates all the feasible events and then calculates the marginal probability of assigning each detection to each track in the cluster.

Update Track Class Probability. Suppose the marginal probabilities of M detections assigned to a track in the cluster are $(\beta_0, \beta_1, \ldots, \beta_M)$, where β_0 is the marginal probability that no measurements is associated with the track. Then the tracker updates the class probability of the track as

$$\mu_k = \left(\sum_{m=1}^{M} \beta_m \frac{c_{j(m)} \otimes \mu(k-1)}{c_{j(m)}^T \mu(k-1)} \right) + \beta_0 \mu(k-1) \tag{8}$$

where $c_j(m)$ is the class probability vector of detection m, $\otimes$ represents element-wise multiplication, $\mu(k-1)$ is the class probability vector of the track in the $k-1$ step, and $\mu(k)$ is the class probability vector of the track in the k step.

The tracker updates the class properties of tracks cluster by cluster.

Process

In this process, we employ a tracker to handle the 2D and 3D bounding boxes obtained from target detection, along with their corresponding timestamps. The tracker utilizes a soft assignment strategy, allowing each trajectory to integrate contributions from multiple detection results. Its responsibilities include trajectory initialization, confirmation, correction, prediction (including prediction in a coasting state without active motion), and deletion. The tracker operates based on the 2D and 3D bounding boxes provided by target detection, as well as the associated timestamps. For each trajectory, the tracker estimates its state vector and the covariance matrix of the state estimation error. It ensures that every detection is assigned to at least one trajectory; if a detection cannot be matched to any existing trajectory, the tracker creates a new one.

Newly generated trajectories are initially in a tentative state. When a tentative trajectory accumulates a sufficient number of detection assignments, its status transitions to confirmed. If the detections themselves already carry explicit classification information (i.e., the returned trajectory data contains non-zero fields), the corresponding trajectory is immediately confirmed as valid. Once a trajectory is confirmed, the tracker recognizes it as representing a real physical object. However, if a trajectory fails to receive any detection assignments within a predefined number of update cycles, it will be deleted.

Through this process, we are ultimately able to obtain the trajectories of vehicles within the intersection.

3.3. Multi intersection and multi-target tracking

We generate vehicles at the same location, placing an RGB camera at the vehicle's starting point and a semantic segmentation camera at the vehicle's endpoint. We capture images from the front and rear directions of the vehicle for each frame, along with the 2D labels.

Then, the vehicle images from the front and rear viewpoints are cropped based on the 2D labels, and images from the same type of vehicle from both viewpoints are combined. Finally, the images are reshaped to a size of 224x224. The imagePretrainedNetwork function is used to train the model, specifying the ResNet50 architecture. This neural network has already learned rich feature representations from a large number of images.

Finally, the vehicle images tracked at Intersection 1 and the images from the corresponding viewpoint camera at Intersection 2 are placed together for re-identification. In other words, the first image is the object to be re-identified, and it will be recognized at the next intersection, thus allowing the integration of both trajectories.

For matching vehicle trajectories between intersections, we currently compute the cosine similarity between all corresponding vehicle trajectories at the two intersections. If Intersection 1 has M vehicles and Intersection 2 has N vehicles, a matrix of size M×N is generated. If the similarity exceeds a certain threshold, the two vehicles are considered to be the same vehicle.

3.4. Trajectory Inference and Reconstruction

Inference of Trajectories in Unknown Regions. Ren et al. proposed an Enhanced Multi-Stream Interaction Network (EMSIN), which utilizes multi-stream inputs (historical trajectories, vehicle interactions, and environmental information) and dynamic interaction modeling (graph neural networks and attention mechanisms) to fuse spatiotemporal features and output the probability distribution of future trajectories, thereby achieving high-precision vehicle trajectory prediction[23]. We have made certain modifications to this framework. In the inputs, for vehicle i, its historical trajectory can be represented as:

$$\mathbf{X}_i = \left[\mathbf{x}_i^{t-T}, \mathbf{x}_i^{t-T+1}, \ldots, \mathbf{x}_i^{t-1}\right] \tag{9}$$

Here, $\mathbf{x}_i^t = (x_i^t, y_i^t)$ represents the position coordinates of vehicle i at time t. The positions and topological structures of the two matched intersections A and B are represented as:

$$\mathbf{R}_A = (x_A, y_A, \text{ Topology }_A) \tag{10}$$

$$\mathbf{R}_B = (x_B, y_B, \text{ Topology }_B) \tag{11}$$

The set of possible paths for a vehicle traveling from intersection A to intersection B is represented as:

$$\mathcal{P}_{A\to B} = \{P_1, P_2, \ldots, P_K\} \tag{12}$$

The formula for predicting vehicle trajectories in unknown regions is:

$$\hat{\mathbf{Y}}_i = \mathbf{W}_h \mathbf{h}_i + \mathbf{W}_r \left(\mathbf{r}_A + \mathbf{r}_B\right) + \mathbf{W}_p \sum_{k=1}^{K} \alpha_k \mathbf{p}_k \tag{13}$$

Where, $\mathbf{W}_h, \mathbf{W}_r, \mathbf{W}_p$ are learnable weight matrices. $\mathbf{h}_i$ represents the historical trajectory encoded using LSTM. $\mathbf{r}_A$ and $\mathbf{r}_A$ are the topological structures of intersections A and B encoded using GNN. $\mathbf{p}_k$ is the path feature extracted after encoding each path P_k. α_k is the attention weight of path P_k, indicating the probability of the vehicle choosing that path.

All Trajectory Restoration. Through the aforementioned network, we are able to obtain vehicle trajectories in unknown regions. Simultaneously, we have also acquired vehicle trajectories at intersections using the JPDA tracker. At this point, we only need to concatenate all trajectories of the same vehicle in chronological order and place them into a new collection, thereby obtaining the complete trajectory of that vehicle. This method can be represented as:

$$S = \sum_{n=1}^{k} (x_t) \tag{14}$$

where S is the trajectory collection, and x_t represents the segmented trajectories labeled with time intervals. By organizing the trajectories of all vehicles in this manner, we can derive the complete vehicle trajectory collection, effectively reconstructing the trajectories of all vehicles.

3.5. Twin of Trajectory

Through the previous steps, we have obtained the trajectories of all vehicles. Here, we will further optimize these trajectories using a trajectory smoothing algorithm, and then employ a PID controller to control the vehicles, ensuring they follow the optimized trajectories[25]. The PID controller consists of three main components: Proportional control (P), Integral control (I), and Derivative control (D). Each component serves a specific purpose, enabling the regulation of different aspects of the system.

- The proportional control component adjusts the control output directly based on the current error. The control output is proportional to the error, meaning that the larger the error, the stronger the control action. The formula is as follows:

$$u_P(t) = K_p \cdot e(t) \tag{15}$$

Here, K_p is the proportional gain, which determines the strength of the proportional control's response to the error.

Proportional control can quickly respond to errors and reduce their magnitude. However, using only proportional control may not completely eliminate steady-state error, which is the small error that remains after the system has stabilized.

- The integral control component accumulates the sum of errors and adjusts the control output based on this accumulated value. The integral term takes into account historical errors and is capable of eliminating steady-state errors. The formula is as follows:

$$u_I(t) = K_i \int_0^t e(\tau) d\tau \tag{16}$$

Here, K_i is the integral gain, which determines the strength of the integral control's response to the accumulated error.

By considering the accumulated error, integral control can eliminate steady-state errors, enabling the system to achieve the target value over the long term. However, excessively high integral gain may lead to system overshoot or oscillation.

- The derivative control component adjusts the control output based on the rate of change of the error. It predicts future error trends and takes preemptive measures. The formula is as follows:

$$u_{\mathrm{D}}(t) = K_d \cdot \frac{de(t)}{dt} \tag{17}$$

Here, K_d is the derivative gain, which determines the strength of the derivative control's response to the rate of change of the error.

Derivative control can suppress rapid changes in errors, reduce system overshoot and oscillation, and make the system response smoother. However, derivative control is sensitive to noise and can amplify high-frequency noise.

The tuning process of the PID controller is as follows: 1. Calculate the error $e(t)$ between the target value (setpoint) and the current system output in real time. 2. Compute the proportional action $K_p \cdot e(t)$ based on the current error, the integral action $K_i \int_0^t e(\tau)d\tau$ based on historical errors, and the derivative action $K_d \cdot \frac{de(t)}{dt}$ based on the rate of change of the error. 3. Sum the results of the proportional, integral, and derivative components to obtain the final control output $u(t)$, which is then applied to the system (for example, adjusting the vehicle's acceleration or direction). 4. The system adjusts based on the control output $u(t)$, changing its state (for example, speed, position, etc.). The controller continues to monitor the system state, updates the error in real time, and repeats the above steps until the system stabilizes near the target value.

3.6. Evaluation

We employed two evaluation metrics.

Multiple Object Tracking Accuracy. Single-camera, multi-object tracking performance is typically measured by the Multiple Object Tracking Accuracy (MOTA):

$$\text{MOTA} = 1 - \frac{FN + FP + M}{T} \tag{18}$$

Among them: FN (False Negatives) represents the number of missed detections, which means targets that truly exist but were not detected. FP (False Positives) represents the number of false detections, which means targets that were detected but do not actually exist. M (Mismatches) represents the number of identity mismatches, which means cases where the tracking ID of a target was incorrectly assigned during the tracking process. T (Total number of true detections) represents the total number of true detections, which means the number of targets correctly detected across all frames.

The value of MOTA ranges from 0 to 1, with higher values indicating better tracking performance. A MOTA of 1 means there are no missed detections, false detections, or identity mismatches, representing perfect tracking performance. A MOTA of 0 means the tracking performance is very poor, with all detections being incorrect.

Multi Camera Tracking Accuracy. The multi-camera object tracking accuracy (MCTA) which condenses all aspects of system performance into one measure[2]:

$$\text{MCTA} = \frac{2PR}{P+R}\left(1 - \frac{M^w}{T^w}\right)\left(1 - \frac{M^h}{T^h}\right) \tag{19}$$

Among them: P represents the proportion of correct matches (correctly identifying and tracking targets) across all cameras. R represents the recall rate, which is the proportion of all true targets that are correctly tracked. M^w represents the number of weak matching errors, which is the number of times a target is incorrectly matched to another target during the tracking process. T^w represents the total number of weak matching attempts, which is the total number of possible matching attempts. M^h represents the number of strong matching errors, which is the number of times a target is incorrectly matched to another target during the tracking process, but here it may refer to a stricter standard for matching errors. T^h represents the total number of strong matching attempts, which is the total number of possible matching attempts.

MCTA evaluates tracking accuracy by considering the proportion of correct matches and the recall rate, while also assessing tracking consistency by penalizing weak and strong matching errors. The higher the MCTA value, the better the performance of the multi-camera tracking system.

4. Experiments

We conducted the following experiments: (a) 3D object detection using PointPillars deep learning for LiDAR, (b) performing multi-target tracking under single-intersection and multi-intersection scenarios, (c) vehicle re-identification between intersections using a ReID network, and (d) implementing trajectory digital twins in Carla.

4.1. preparation

Simulation Environment. We conducted simulation experiments in Carla, whose advantage lies in its provision of a highly realistic virtual environment capable of accurately simulating complex traffic scenarios and diverse driving conditions. Carla supports flexible sensor configurations, such as cameras and LiDAR, facilitating multimodal data acquisition and fusion experiments[18]. We selected Town10 and Town1 in Carla as our simulation scenarios, which offer the benefit of providing highly realistic and diverse virtual environments, enabling precise simulation and replication of real traffic scenarios. The complex urban structure and dense traffic flow of Town10 can simulate highly dynamic real traffic environments, while the simple layout and clear rules of Town1 make it easy to construct controlled experimental scenarios.

Data Collection. In CARLA's Town10 scenario, a single intersection location is selected, with a LiDAR placed at the center and six cameras arranged around it to achieve 360-degree omnidirectional perception coverage. This setup allows for the acquisition of rich point cloud data and image information, making it suitable for target recognition and tracking in complex traffic environments. By configuring the LiDAR with 64 channels, a detection range of 100 meters, 250,000 points per second, and a rotation frequency

of 20 Hz, the radar can efficiently generate high-density and accurate point cloud data. Additionally, one camera is placed at the front and rear of the vehicle, and one camera is positioned on each of the four roads to the left and right. The cameras have a resolution of 1920x1080 pixels and a field of view of 90 degrees, ensuring the capture of wide-angle image data in high definition. The collected data includes point cloud data from the LiDAR and image data from the cameras. The radar data is saved as MAT files with timestamps, while the camera data is saved as images for each frame in six directions. The data structure follows the storage format of the Panda dataset, making the data more standardized for subsequent processing and fusion[22]. Each frame of radar data stores information such as point cloud objects, timestamps, positions relative to the ego vehicle, and detected 3D bounding boxes. Each frame of camera data stores image data from six directions, positions relative to the ego vehicle, timestamps, and detected 2D bounding boxes.

4.2. Experimental effect

4.3. Details Explanation

4.4. Limitations and Future Directions

Problems Encountered in The Experiment. During our experiments, we encountered the following issues: 1.When matching trajectories across multiple intersections, images of vehicles corresponding to each trajectory are needed to extract re-identification appearance features. Acquiring these images is difficult, and the methods used can affect accuracy. 2.When matching trajectories, it is necessary to associate trajectories from all intersections. However, due to ID switching, the trajectory of the same vehicle at a single intersection may be fragmented, making integration complex. If only one trajectory is selected as the current intersection's trajectory for a vehicle, issues arise when the same vehicle returns to the intersection.

Shortcomings of Current Research. Although our experiments have achieved some success, there are still some limitations. For example, the detection accuracy of PointPillars is not high, resulting in suboptimal tracking performance. Additionally, false detections may occur when acquiring images of vehicles corresponding to current trajectories, potentially leading to trajectory matching errors.

Vision of The Future. Our experiments were conducted under offline conditions, meaning they lacked real-time capabilities. In future work, we aim to transition these experiments to an online framework to achieve high real-time performance, thereby enhancing their experimental value.

5. Conclusion

Through a series of experiments, we have demonstrated the application value of digital twins in intelligent transportation. Digital twin technology can not only accurately replicate real-world traffic scenarios but also effectively predict future traffic conditions. Multi-target tracking and the inference of unknown trajectories serve as strong evidence of this capability. This holds significant value for the development of future cities.

We hope that new large-scale datasets will be introduced in the future to further enhance the effectiveness of digital twins.

References

[1] Sadeq Almeaibed, Saba Al-Rubaye, Antonios Tsourdos, and Nicolas P Avdelidis. Digital twin analysis to promote safety and security in autonomous vehicles. *IEEE Communications Standards Magazine*, 5(1):40–46, 2021. 1

[2] Temitope Ibrahim Amosa, Patrick Sebastian, Lila Iznita Izhar, Oladimeji Ibrahim, Lukman Shehu Ayinla, Abdulrahman Abdullah Bahashwan, Abubakar Bala, and Yau Alhaji Samaila. Multi-camera multi-object tracking: A review of current trends and future advances. *Neurocomputing*, 552: 126558, 2023. 6

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1

[4] Manuel Dominguez-Rodrigo, Ander Fernandez-Jauregui, Gabriel Cifuentes-Alcobendas, and Enrique Baquedano. Use of generative adversarial networks (gan) for taphonomic image augmentation and model protocol for the deep learning analysis of bone surface modifications. *Applied Sciences*, 11 (11):5237, 2021. 3

[5] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary perspectives on complex systems: New findings and approaches*, pages 85–113, 2017. 1

[6] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary perspectives on complex systems: New findings and approaches*, pages 85–113, 2017. 1

[7] Bing He, Jia Li, Yifan Zhao, and Yonghong Tian. Part-regularized near-duplicate vehicle re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3997–4005, 2019. 2

[8] Yaqi Hu, Maoqiang Wu, Jiawen Kang, and Rong Yu. D-tracking: digital twin enabled trajectory tracking system of autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 2024. 2

[9] Zhongxu Hu, Shanhe Lou, Yang Xing, Xiao Wang, Dongpu Cao, and Chen Lv. Review and perspectives on driver digital twin and its enabling technologies for intelligent vehicles. *IEEE Transactions on Intelligent Vehicles*, 7(3):417–440, 2022. 1

[10] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP journal of manufacturing science and technology*, 29:36–52, 2020. 1

[11] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of

deep convolutional neural networks. *Artificial intelligence review*, 53:5455–5516, 2020. 2

[12] Pirazh Khorramshahi, Amit Kumar, Neehar Peri, Sai Saketh Rambhatla, Jun-Cheng Chen, and Rama Chellappa. A dual-path model with adaptive attention for vehicle re-identification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6132–6141, 2019. 2

[13] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 2

[14] Yu-Jhe Li, Jinhyung Park, Matthew O'Toole, and Kris Kitani. Modality-agnostic learning for radar-lidar fusion in vehicle detection. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 918–927, 2022. 2

[15] Xiwen Liao, Supeng Leng, Yao Sun, Ke Zhang, and Muhammad Ali Imran. A digital twin-based traffic guidance scheme for autonomous driving. *IEEE Internet of Things Journal*, 2024. 1

[16] Zheng Liu, George Xiao, Huan Liu, and Hanbing Wei. Multi-sensor measurement and data fusion. *IEEE Instrumentation & Measurement Magazine*, 25(1):28–36, 2022. 3

[17] Yuqian Lu, Chao Liu, I Kevin, Kai Wang, Huiyue Huang, and Xun Xu. Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and computer-integrated manufacturing*, 61: 101837, 2020. 1

[18] Sumbal Malik, Manzoor Ahmed Khan, and Hesham El-Sayed. Carla: Car learning to act—an inside out. *Procedia Computer Science*, 198:742–749, 2022. 6

[19] Dimitrios Meimetis, Ioannis Daramouskas, Isidoros Perikos, and Ioannis Hatzilygeroudis. Real-time multiple object tracking using deep learning methods. *Neural Computing and Applications*, 35(1):89–118, 2023. 2

[20] Kaleb Ben Naveed, Zhiqian Qiao, and John M Dolan. Trajectory planning for autonomous vehicles using hierarchical reinforcement learning. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 601–606. IEEE, 2021. 3

[21] Óscar Pérez-Gil, Rafael Barea, Elena López-Guillén, Luis M Bergasa, Carlos Gómez-Huélamo, Rodrigo Gutiérrez, and Alejandro Díaz-Díaz. Deep reinforcement learning based control for autonomous vehicles in carla. *Multimedia Tools and Applications*, 81(3):3553–3576, 2022. 3

[22] Tal Reiss, Niv Cohen, Liron Bergman, and Yedid Hoshen. Panda: Adapting pretrained features for anomaly detection and segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2806–2814, 2021. 7

[23] Yilong Ren, Zhengxing Lan, Lingshan Liu, and Haiyang Yu. Emsin: Enhanced multistream interaction network for vehicle trajectory prediction. *IEEE Transactions on Fuzzy Systems*, 33(1):54–68, 2024. 5

[24] Muhammad Usman Shoukat, Lirong Yan, Yukai Yan, Fan Zhang, Yikang Zhai, Peng Han, Saqib Ali Nawaz, Muhammad Ahmad Raza, Muhammad Waqas Akbar, and Abid Hussain. Autonomous driving test system under hybrid reality: The role of digital twin technology. *Internet of Things*, 27: 101301, 2024. 1

[25] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z Morley Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15159–15168, 2022. 5

[26] 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 *European conference on computer vision*, pages 1–21. Springer, 2022. 2

[27] Xiangmo Zhao, Pengpeng Sun, Zhigang Xu, Haigen Min, and Hongkai Yu. Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications. *IEEE Sensors Journal*, 20(9):4901–4913, 2020. 2

[28] Zhedong Zheng, Tao Ruan, Yunchao Wei, Yi Yang, and Tao Mei. Vehiclenet: Learning robust visual representation for vehicle re-identification. *IEEE Transactions on Multimedia*, 23:2683–2693, 2020. 2