2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) | 978-1-6654-6499-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/MCSOC57363.2022.00026

Radar and Camera Fusion for Object Forecasting in Driving Scenarios

Albert Budi Christian, Yu-Hsuan Wu, Chih-Yu Lin*, Lan-Da Van, and Yu-Chee Tseng
Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan
*Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan
Email: albert.c@nycu.edu.tw, t22170000@gmail.com, lincyu@mail.ntou.edu.tw,
ldvan@cs.nycu.edu.tw, yctseng@cs.nycu.edu.tw

***Abstract*—In this paper, we propose a sensor fusion architecture that combines data collected by the camera and radars and utilizes radar velocity for road users' trajectory prediction in real-world driving scenarios. This architecture is multi-stage, following the detect-track-predict paradigm. In the detection stage, camera images and radar point clouds are used to detect objects in the vehicle's surroundings by adopting two object detection models. The detected objects are tracked by an online tracking method. We also design a radar association method to extract radar velocity for an object. In the prediction stage, we build a recurrent neural network to process an object's temporal sequence of positions and velocities and predict future trajectories. Experiments on the real-world autonomous driving nuScenes dataset show that the radar velocity mainly affects the center of the bounding box representing the position of an object and thus improves the prediction performance.**

***Index Terms*—Camera, data fusion, object forecasting, radar, trajectory prediction, velocity**

I. INTRODUCTION

Trajectory prediction or object forecasting plays a significant role in an autonomous driving system and the corresponding scenarios. By predicting where the road users may move in the future, an autonomous vehicle can plan a safe path or make the driving decision to avoid potential dangers such as collisions with vulnerable road users [1], [2]. For example, if pedestrians or cyclists cross the street, passing vehicles should slow down to prevent harmful events. Various traffic situations make it necessary to predict future positions to ensure safety. For solving the trajectory prediction problem, typical approaches [3]–[7] adopt the detect-track-predict paradigm. Road users are first detected using sensor data such as camera images and point clouds from LiDARs or radars. The detected objects at each time-step are then tracked to produce sequences of positions used in prediction. By dividing this issue into separate stages, the framework's design is flexible as each module can focus on a single task.

On the other hand, different sensor data are often combined to give more reliable information about the dynamic environment. Sensors such as cameras, LiDARs, and radars on a self-driving car have distinct characteristics. The cameras provide rich visual features, and the LiDARs give the 3D shapes of objects based on accurate depth information. The radars can measure objects' distance and velocity at low resolution. The complementary properties of these three sensors motivate researchers to study sensor fusion for autonomous driving tasks, especially object detection [8]. However, research on fusion for the trajectory prediction problem is still lacking.

In this paper, we focus on the fusion between camera and radars. The objective is to investigate whether radar velocities could enhance the performance of trajectory prediction on the image plane. To this end, we design a sensor fusion architecture that combines images and point clouds to address this problem. We employ the detect-track-predict paradigm and use an independent module for each stage. First, we adopt two object detectors and merge detection results from these two models using the affirmative strategy introduced in [9] to locate objects on images. Detection for the same object are then associated between frames. The prediction module in the last stage generates a sequence of locations over a while in the future. In addition, we insert a radar association module between the tracking module and the prediction module for associating tracked objects with their radar velocities. This additional information will also be used in prediction. We evaluate our design on nuScenes [10], a dataset published for posing various autonomous driving problems. We show that radar velocity can improve prediction by comparing results between models with and without using radar velocity.

The rest of this paper is organized as follows. Related works from detection to prediction are reviewed in Section II. The proposed fusion architecture is described in Section III. Experiment results, including performance comparisons and prediction examples, are presented in Section IV, and the conclusion is given in Section V.

II. RELATED WORKS

In this section, we introduce the researches for perception and prediction problems in autonomous driving.

A. Deep 2D Multi-modal Object Detection

In recent research, many works have focused on multi-modal object detection for autonomous driving using deep learning models. Most of the existing methods for 2D detection discuss the fusion between LiDARs and cameras. These approaches use either one-stage detectors or two-stage detectors to detect road users. For example, Asvadi et al. [11] use multiple YOLOv2 [12] models to detect vehicles using LiDAR or camera data separately and fuse detections from these models. In [13], Regions of Interest (RoIs) are extracted from RGB images and LiDAR data and then fed into the Fast R-CNN network to detect pedestrians and cyclists. In [14], LiDAR points are downsampled and clustered to generate region proposals, which are input into a convolutional neural network to detect pedestrians. Since camera and radar data sensing modalities are complementary in human sensing, recent research have shown that combining these two sensing modalities significantly improves performance in object detection, human identity, and localization [15], [16]. [17] applies camera and radar fusion task to detect pedestrian. Previous work [18] utilizes two branches of neural networks to extract and concatenate feature maps extracted from radar and pictures. [19] proposes an object detection system based on sensor fusion that employs a camera and mmWave radar. Radar Region Proposal Network (RRPN) [20] adopts a two-stage detector structure and exploits radar points without clustering them in the region proposal stage. RRPN [20] will be utilized in our work for the detection of road users, and additional details will be presented.

B. Multiple Object Tracking

The target of Multiple Object Tracking (MOT) is to associate objects of the same identity across consecutive frames. Most MOT approaches and architectures adopt the tracking-by-detection paradigm. Emphasizing the importance of high detection quality, Simple Online and Realtime Tracking (SORT) [21] makes use of the Kalman filter [22] and Hungarian algorithm [23] to deal with MOT. Wojke et al. [24] extends SORT by incorporating the visual appearance of tracked objects. In our work, we choose to use SORT [21] due to its speed and accuracy.

C. Trajectory Prediction

The trajectory prediction task can be treated as a sequence-to-sequence problem, often addressed by the encoder-decoder architecture built with the recurrent neural network (RNN) or its variants. The encoder-decoder architecture was previously proposed for Natural Language Processing (NLP) tasks such as machine translation [25]–[27]. The encoder encodes the input sequence into a state vector, which is then decoded by the decoder to produce the output sequence. This architecture has also been used for trajectory prediction afterward [3]–[7]. Among these works, most use an RNN variant called Long Short-Term Memory (LSTM) to process sequential data. Park et al. [4] used an LSTM encoder-decoder and applied the beam-search algorithm to produce the most probable future trajectories of surrounding vehicles. In [6], a convolutional social pooling layer is proposed to maintain spatial information and learn vehicle interactions from the outputs of the LSTM encoder. An LSTM decoder is then used to predict vehicle motion. On the other hand, the predicted trajectories can be represented in many forms, such as on the occupancy grid map (OGM), which uses the grid index to indicate a location. Another form is to place the predictions on the first-person footage or on-board camera images. This representation is used in Spatio-Temporal Encoder-Decoder (STED) [7]. STED [7] computes the optical flow for motion feature extraction using FlowNet2 [28] and utilizes these features in the encoder-decoder for trajectory prediction. Our work follows the problem formulation presented in [7] and adopts the encoder-decoder model architecture [7], [27] for trajectory prediction. As a result, we will compare the performances of our system model and STED [7] later in the experiment. In summary, our work builds on previous research to explore the effect of radar data on the performance of trajectory prediction.

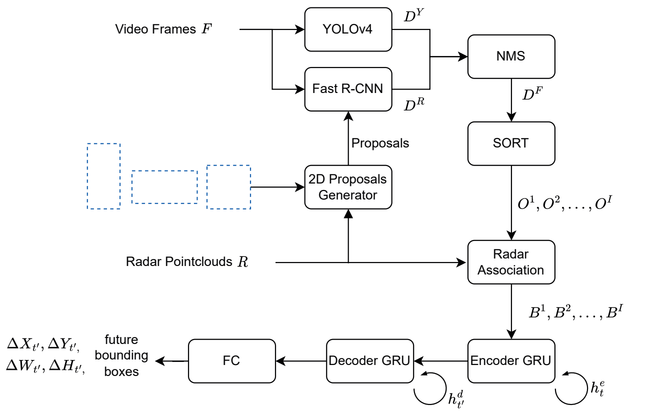

Fig. 1: Proposed system model architecture.

III. System Model

In this section, we present our architecture for sensor fusion between camera and radars mounted on a moving vehicle. Given a sequence of m video frames $F = F_{t-m+1}, ..., F_t$ and m radar point clouds $R = R_{t-m+1}, ..., R_t$ in the past one second, our goal is to detect the surrounding objects in these frames and predict their future locations for the next two seconds. The proposed system model architecture includes three main phases: object detection phase, tracking phase, and prediction phase, as shown in Fig. 1. In the following, we will describe these three phases in details.

A. Object Detection

The first stage for future bounding box prediction is to know what objects are in the surroundings at each timestep. To handle a variety of scenarios, we use two object detectors: YOLOv4 [29] and RRPN [20]. RRPN is a proposal generator with Fast R-CNN [30]. Then we combine these detection results by applying Non-Maximum Suppression (NMS).

1) YOLOv4: One of the object detectors we use in the architecture is YOLOv4. YOLOv4 is a high-accuracy, one-stage object detection network which is fast and suitable for real-time applications such as self-driving cars. The network architecture of YOLOv4 is composed of three parts: backbone, neck, and head. The input is an image. The backbone is responsible for extracting levels of features from the input image. The additional layers between the backbone and the head are called the neck. These layers are used to combine the feature maps of different layers in the backbone and can be used to detect objects at different scales. The head part is then applied to predict bounding boxes. In the training stage, we use pre-trained weights for the convolutional layers of the YOLOv4 network and further train the model on the nuScenes dataset [10]. In order to obtain higher precision, the network resolution is increased to 608 by 608.

After training, we apply the model to each frame in F. The frame at timestep t is denoted as F_t. For each timestep t, the model takes F_t as input and performs object detection on it to output a set of detected objects D_t^Y. Each bounding box in D_t^Y is denoted as $(x, y, w, h, class, conf)$ where x and y are the coordinates of the center, w and h are the width and the height of the bounding box, respectively, $class$ is the class of the object, and $conf$ is the confidence of the prediction. Our customized YOLOv4 model is trained to detect objects for the following categories: pedestrian, bicycle, car, and motorcycle.

2) Proposals Generator and Fast R-CNN: The second object detector used in our work is RRPN [20] that emphasizes parts of the image where radar detections exist. The main idea of this module is to perform object detection with radar information. RRPN [20] has two components: a proposals generator and a detection network. The proposals generator is an object proposal method based on radar detections. The detection network, Fast R-CNN, is a two-stage object detection algorithm that first uses some techniques to generate several candidate object regions and then classifies these regions and refines their scales and positions using a convolutional neural network. In RRPN, the radar-based proposals generator is used to generate these regions instead of using the computationally expensive and time-consuming Selective Search algorithm [31] originally adopted in Fast R-CNN. More details about these components are described below.

Proposals Generator This component takes a radar point cloud R_t and multiple anchor boxes as input and outputs a set of proposals corresponding to the frame F_t for each timestep t. A radar point cloud is a set of radar points. Each radar point is expressed as $(x_{raw}, y_{raw}, z_{raw}, v_{raw})$, representing the raw 3D coordinates and the velocity in radar coordinate system. The anchor boxes introduced in Faster R-CNN [32] are a set of pre-defined rectangular regions with different sizes and aspect ratios. Following [20], we use four sizes and three aspect ratios, resulting in a total of twelve different anchor boxes. To locate the region proposals, first, we map each raw radar point in R_t from the sensor coordinate system to the image coordinate system using the translation and rotation matrices. Each mapped radar point is in the format (x, y, d), where (x, y) is the coordinate on the image plane and d is the depth. Second, each mapped point serves as the center of the twelve anchor boxes so that a set of regions is determined. Lastly, these regions are scaled based on the depth of the radar point to get final object proposals.

Fast R-CNN Fast R-CNN takes the frame F_t and the object proposals for this frame as input. It first produces feature maps for the whole image, and the pre-computed proposals are projected onto the feature maps to get the mapped RoIs. An RoI pooling layer is then applied to unify the dimension of these mapped RoIs. Finally, a fully connected network is used to generate the results, with a pair of output layers: one for object classification and the other for bounding box regression. For the frame F_t, we obtain a set of detections D_t^R.

3) Non-Maximum Suppression: A vehicle or a pedestrian may be detected by both YOLOv4 and RRPN, which can result in too many unnecessary bounding boxes. Therefore, we use the NMS algorithm to remove redundant bounding boxes after obtaining the detections, D_t^Y and D_t^R, from the two detectors. NMS is a post-processing method that decides which detections to keep based on a threshold value, *iou_thresh*. In this work, we set *iou_thresh* to 0.5. NMS operates as follows:

Step 1: Make two sets: the union of two detection sets, D_t^Y and D_t^R, denoted as D_t^U , and the final detection set denoted as D_t^F. D_t^F is initialized to an empty set.

Step 2: Select the bounding box with the highest confidence score from the set D_t^U and add it to D_t^F.

Step 3: Calculate the Intersection over Union (IOU) between every other bounding box of the same class in D_t^U and this best box. If the IOU is greater than *iou_thresh*, the box is removed from D_t^U.

Step 4: Remove the best bounding box from D_t^U.

Steps 2, 3, and 4 are repeated until D_t^U becomes empty. NMS is applied to $(D_t^Y \cup D_t^R)$ and outputs the set of final detections, D_t^F, in the frame F_t for each timestep t.

B. Tracking

To predict the future locations of an object, we need to know the sequence of past positions of this object first, which means that we need a tracker to identify which bounding box belongs to the same object across all frames. For this purpose, we use the SORT algorithm [21].

After detecting objects for each frame in F, we obtain a sequence of m sets of detections $D^F = D_{t-m+1}^F, ..., D_t^F$. Note that we do not specify the number of detected objects in each D_t^F at timestep t since this number may vary from frame to frame. The sequence D^F is then fed into SORT for object tracking.

In the following, we describe how we use SORT:

Step 1: Use a tracker to keep track of an individual object. The tracker keeps a record of the past bounding boxes of its tracked object.

Step 2: Estimate the positions in the current frame of the existing tracked objects based on past trajectories.

Step 3: Measure the IOU between each detection and all estimations of the current frame. Assign the detection to the existing trackers by maximizing the total IOU.

Step 4: Update the tracker by adding the assigned detection to it.

Step 5: Create a new tracker and a new identity for detection not assigned to any existing tracker.

Step 6: Repeat steps 2 to 5 m times for all detection in D^F to get the tracking result.

Finally, we get a list of tracked objects with varying lengths of bounding box history as there may be objects leaving the image in earlier frames or entering in later frames. For sub-sequent predictions, we discard tracks shorter than m. The remaining tracks are denoted as $O^1, O^2, ..., O^I$ where I is the number of fully tracked objects. $O^i = O^i_{t-m+1}, O^i_{t-m+2}, ..., O^i_t$ is the sequence of the observations of object i in the past m frames where $O^i_t = (x^i_t, y^i_t, w^i_t, h^i_t)$ represents the bounding box of object i at timestep t.

C. Radar Association

Following the tracking phase, we add the radar association module. This module aims at using instantaneous velocity data of radar points to help prediction. To associate the radar points with the detected objects, first, the radar points are projected onto the image plane using the same matrices mentioned in Section III-A2. In addition, we also transform the velocity of each point. After obtaining the image coordinates of each radar point, we choose the one inside the bounding box and consider it the radar detection of the object. If there are multiple points inside the box, we select the one with minimum depth. Having the radar detection for an object, we then take its velocity as the extra feature for prediction.

For each bounding box of each tracked object i at each timestep t, we use the above procedures to find the radar detection and concatenate the velocity to the corresponding O^i_t. Therefore, for each object i, we have another sequence of information, which we denote by B^i, about it. $B^i = B^i_{t-m+1}, B^i_{t-m+2}, ..., B^i_t$ is the sequence of combinations of the position and velocity. Each B^i_t has the format $(x^i_t, y^i_t, w^i_t, h^i_t, v^i_t)$, where v^i_t is the velocity of object i at timestep t. Finally, each B^i is used as input to the prediction model.

D. Prediction

In this stage, we employ an encoder-decoder architecture [7], [27] based on Gated Recurrent Unit (GRU) [26], [27], a variant of recurrent neural network, to predict future n bounding boxes using the past $m(boundingbox, velocity)$ pairs of an object. In this work, m and n are set to 12 and 24, respectively, corresponding to one second and two seconds. Fig. 2 shows the internal prediction model architecture. The encoder and the decoder are trained end-to-end using Smooth L1 as the loss function. The major difference between the model architectures in Fig. 2 and [7] is that the convolutional neural network for optical flow is excluded in Fig. 2.

1) Encoder GRU: The encoder, which is constructed by multiple GRU cells, is responsible for giving a summary of past information about a single object in the form of a fixed-length feature vector. For each past timestep t, the GRU cell takes B^i_t along with previous hidden state h^e_{t-1} as input and uses these data to update the 256-dimensional hidden state vector, which is passed down to the next time-step. After processing every element in the sequence B^i, the encoder outputs the hidden state at the last time-step h^e_t. This state vector is then passed to the decoder.

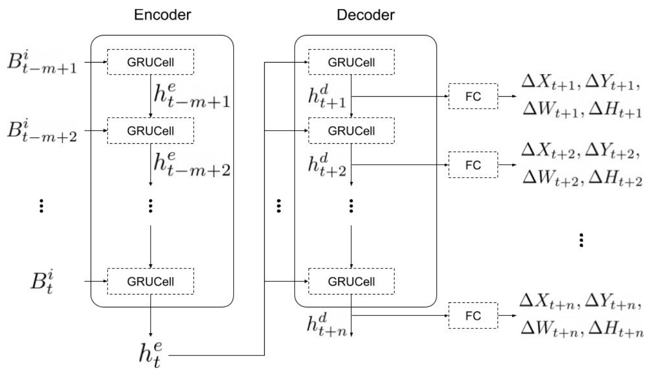
Encoder
Decoder
B^i_{t-m+1}
B^i_{t-m+2}
B^i_t
GRUCell
h^e_{t-m+1}
h^e_{t-m+2}
h^e_t
h^d_{t+1}
h^d_{t+2}
h^d_{t+n}
FC
$\Delta X_{t+1}, \Delta Y_{t+1}, \Delta W_{t+1}, \Delta H_{t+1}$
$\Delta X_{t+2}, \Delta Y_{t+2}, \Delta W_{t+2}, \Delta H_{t+2}$
$\Delta X_{t+n}, \Delta Y_{t+n}, \Delta W_{t+n}, \Delta H_{t+n}$

Fig. 2: Internal prediction model architecture.

2) Decoder GRU and Fully Connected Layer: Having the vector representation of the encoded position and velocity history of an object, we use a decoder to generate the sequence of predictions. The decoder consists of n pairs of the GRU cell and the fully connected layer. For each future time-step t', the state vector h^e_t and the previous hidden state $h^d_{t'-1}$ are fed into the GRU cell to produce the current hidden state $h^d_{t'}$. The fully connected layer then takes this state vector as input and outputs a 4-dimensional bounding box prediction $(\triangle X_{t'}, \triangle Y_{t'}, \triangle W_{t'}, \triangle H_{t'})$, representing the horizontal and vertical pixel offsets of the center, and the change in width and height of the predicted bounding box on the image plane. Each pair outputs the prediction for a single future timestep. Ultimately, we get n predictions for the future two seconds.

IV. EXPERIMENTAL RESULTS

A. Dataset

We use nuScenes [10], a public autonomous driving dataset, to evaluate our system model. The nuScenes dataset provides hundreds of driving scenes captured by various sensors such as LiDAR, camera, and radar in cities with heavy traffic. Each scene is twenty seconds long. The vehicle used to collect data is equipped with six cameras and five radars. In our experiment, data from one front camera and three front radars [33] are fused as there is an overlap between the Field of Views (FOVs) of each radar and the front camera. The capture frequency of the camera is 12Hz while that of the radar is 13Hz. As the bounding box predictions are on video frames, we select the radar sweep of each of these three radars closest to the time each video frame was captured to synchronize data from different sensors for a single timestep.

We split the dataset into training and test sets with an 8:2 ratio based on the number of scenes. Because we use past one-second sensor data to make future two-second predictions, we need to ensure that an object is visible at least for 3 seconds for prediction performance evaluation. Thus, we further find out the samples that meet the requirements in each scene. Finally, we have a total of 764,825 samples in the dataset.

B. Metrics

For evaluating the performance of bounding box predictions for a single object, we use four metrics: Average/Final

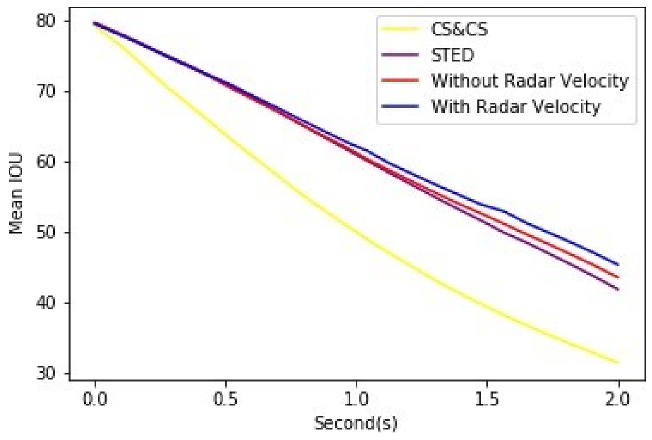

Fig. 3: Mean IOU Change.

Displacement Error (ADE/FDE) and Average/Final Intersection over Union (AIOU/FIOU). DE measures the Euclidean distance between the two centers of the predicted bounding box and the ground truth bounding box. On the other hand, IOU measures the overlapping degree between two bounding boxes. Equations (1), (2), (3), (4) shows how these metrics are calculated.

$$ADE = \frac{1}{n} \sum_{t'=t+1}^{t+n} \parallel c_{t'} - \hat{c}_{t'} \parallel^2 \quad (1)$$

$$FDE = \parallel c_{t+n} - \hat{c}_{t+n} \parallel^2 \quad (2)$$

$$AIOU = \frac{1}{n} \sum_{t'=t+1}^{t+n} \frac{Area(p_{t'} \cap \hat{p}_{t'})}{Area(p_{t'} \cup \hat{p}_{t'})} \quad (3)$$

$$FIOU = \frac{Area(p_{t+n} \cap \hat{p}_{t+n})}{Area(p_{t+n} \cup \hat{p}_{t+n})} \quad (4)$$

whereas, $c_{t'}$ is the center of the predicted bounding box $p_{t'}$ and $\hat{c}_{t'}$ is the center of the ground truth bounding box $\hat{p}_{t'}$ at time-step t'. These metrics are calculated for the objects whose m detected bounding boxes are all true positives. We do not take detection that is false positive or false negative into account when calculating these metrics as there are no ground truth data for the former and no past positions of the latter.

C. Baseline

We take the Constant Shift-Constant Scale (CS & CS) model as the baseline model. In trajectory prediction literature [7], [34], [35], this model is a widely-used baseline method that assumes that the bounding box of an object moves with uniform box velocity on the image plane and does not change its width and height over time. In this work, the box velocity is computed by taking the last two detected positions of the object. Then we can use this velocity to locate the bounding box for each future time-step.

TABLE I: Mean performance of nuScenes test data.

Model	Radar Velocity	AIOU ↗	FIOU ↗	ADE ↙	FDE ↙
CS & CS	-	52.0	31.3	46.6	96.3
STED [7]	-	61.0	41.8	37.5	77.5
Our model	✗	61.4	43.5	38.0	78.4
Our model	✓	62.3	45.3	36.5	74.1

D. Results

To evaluate the effectiveness of our proposed system model in Fig. 1 and the impact of radar velocities, we compare STED [7] and our system model without using velocity data in addition to the CS & CS model. Table I shows the mean performance of our test data. AIOU and FIOU are given in percentage, while ADE and FDE are given in pixels. ↗ indicates that the higher the value, the better the model's performance. Otherwise, ↙ indicates that the lower the value, the better the model's performance. Our model with velocities yields the best performance across all metrics, showing that the radar velocity is effective for the predictions. However, the IOU metrics do not improve so much as the radar velocities do not help predict the width and height changes of the bounding boxes. We also present mean IOU and mean DE during future two seconds. Fig. 3 and 4 show the comparisons between the four models. Using radar velocities, our model keeps the best IOU and DE for almost all time-steps.

Fig. 5 shows some examples of the prediction results. Each row is a sample from the test set. The first column visualizes the trajectory of the center of the predicted bounding box from 0 to 2 seconds in the future. The second and third columns display the predicted bounding boxes at future 1s and 2s, respectively. Our model with radar velocity works well at both night and day.

V. CONCLUSION AND FUTURE WORKS

In this work, we propose a system model architecture fusing different sensor data including radar point clouds and camera images for object forecasting in autonomous driving. We adopt

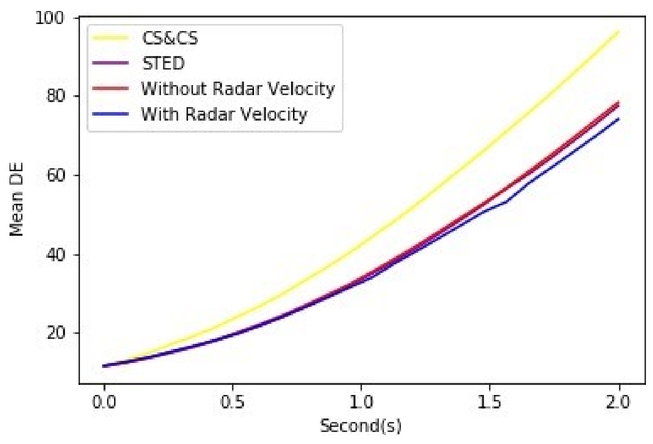

Fig. 4: Mean Displacement Error Change.

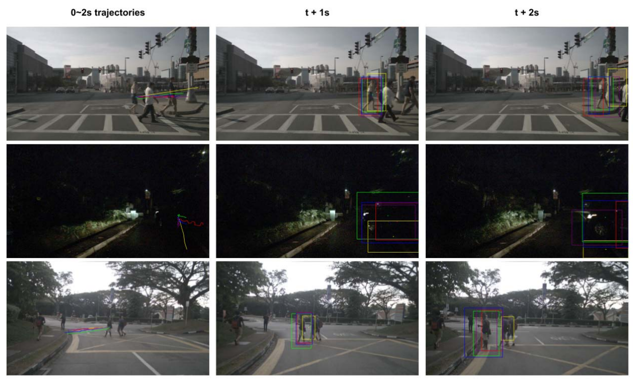

Fig. 5: Bounding box predictions examples. Different models are represented using different colors: ground truth (green), CS&CS (yellow), STED (purple), without radar velocity (red), and with radar velocity (blue).

two object detection models and a tracking algorithm to build the sequence of past positions of the road users. To use radar velocity to assist in forecasting, we propose a radar association method that finds the radar measurement for an object. Based on the encoder-decoder architecture, we can predict object positions for two seconds in the future. The experiment results on the nuScenes dataset show that the performance of trajectory prediction becomes better by adding radar velocities. In the future, we may explore the proposed system model's performance and limitation on other public datasets.

ACKNOWLEDGMENT

This work is supported by National Science and Technology Council, R.O.C under 110-2221-E-019-040-MY3 and PAIR LABs.

REFERENCES

[1] P. Mannion, "Vulnerable Road User Detection: State-of-the-Art and Open Challenges," *arXiv preprint arXiv:1902.03601*, 2019.

[2] A. Ranga, F. Giruzzi, J. Bhanushali, E. Wirbel, P. Pérez, T.-H. Vu, and X. Perotton, "VRUNet: Multi-Task Learning Model for Intent Prediction of Vulnerable Road Users," *Electronic Imaging*, vol. 2020, no. 16, pp. 109–1, 2020.

[3] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes With Interacting Agents," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[4] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1672–1678.

[5] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories With Generative Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[6] N. Deo and M. M. Trivedi, "Convolutional Social Pooling for Vehicle Trajectory Prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[7] O. Styles, V. Sanchez, and T. Guha, "Multiple Object Forecasting: Predicting Future Object Locations in Diverse Environments," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 690–699.

[8] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.

[9] Á. Casado-García and J. Heras, "Ensemble Methods for Object Detection," in *ECAI 2020*. IOS Press, 2020, pp. 2688–2695.

[10] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[11] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "Multimodal Vehicle Detection: Fusing 3D-LIDAR and Color Camera Data," *Pattern Recognition Letters*, vol. 115, pp. 20–29, 2018.

[12] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[13] T. Kim and J. Ghosh, "Robust Detection of Non-motorized Road Users using Deep Learning on Optical and LiDAR Data," in *Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 271–276.

[14] D. Matti, H. K. Ekenel, and J.-P. Thiran, "Combining LiDAR Space Clustering and Convolutional Neural Networks for Pedestrian Detection," in *Proceedings of the IEEE 14th International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[15] K. Qian, S. Zhu, X. Zhang, and L. E. Li, "Robust Multimodal Vehicle Detection in Foggy Weather Using Complementary Lidar and Radar Signals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 444–453.

[16] X. Tang, Z. Zhang, and Y. Qin, "On-Road Object Detection and Tracking Based on Radar and Vision Fusion: A Review," *IEEE Intelligent Transportation Systems Magazine*, 2021.

[17] A. Palffy, J. F. Kooij, and D. M. Gavrila, "Occlusion Aware Sensor Fusion for Early Crossing Pedestrian Setection," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1768–1774.

[18] J. Zhang, M. Zhang, Z. Fang, Y. Wang, X. Zhao, and S. Pu, "RVDet: Feature-level Fusion of Radar and Camera for Object Detection," in *Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2822–2828.

[19] X. Shuai, Y. Shen, Y. Tang, S. Shi, L. Ji, and G. Xing, "milliEye: A Lightweight mmWave Radar and Camera Fusion System for Robust Object Detection," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 145–157.

[20] R. Nabati and H. Qi, "RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 3093–3097.

[21] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime tracking," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

[22] R. E. Kalman *et al.*, "A New Approach to Linear Filtering and Prediction

Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[23] H. Kuhn, "The Hungarian Method for the Assignment and Transportation Problems," *Management Science,*, vol. 10, no. 3, pp. 578–593, 1964.

[24] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649.

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[26] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014.

[28] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[29] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[30] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[31] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[33] R. Nabati and H. Qi, "Centerfusion: Center-Based Radar and Camera Fusion for 3D Object Detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1527–1536.

[34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[35] 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.