P2SIF: Sensor Fusion for Autonomous Driving

Pushkal Mishra
University of California San Diego
United States
pumishra@ucsd.edu

Abstract

P2SIF: Plug-and-Play Sensor Integration Framework is a platform that enhances autonomous driving by integrating specialized models for LiDAR and Radar captioning. Using data from the CARLA simulator, CLIP-based models were trained to generate textual scene descriptions from sensor inputs. While Radar-based captioning proves highly accurate, LiDAR struggles due to 2D projection losses. The results highlight Radar's potential for robust perception and the need for improved LiDAR processing. This work advances the integration of LLMs into autonomous driving for enhanced decision-making.

CCS Concepts

• **Computing methodologies** → **Machine learning**; **Computer vision**; **Modeling and simulation**.

ACM Reference Format:
Pushkal Mishra. 2025. P2SIF: Sensor Fusion for Autonomous Driving. In *The 23rd ACM Conference on Embedded Networked Sensor Systems (SenSys '25), May 6–9, 2025, Irvine, CA, USA.* ACM, Irvine, CA, USA, 7 pages. https://doi.org/10.1145/3715014.3724379

1 Introduction

Large Language Models (LLMs), a key development in the field of Artificial Intelligence (AI), have demonstrated advanced reasoning capabilities and an understanding of complex logic [1, 2]. Many applications across a plethora of disciplines benefit from these advanced reasoning capabilities, and with the exponential growth in their size and performance, LLMs are poised to revolutionize numerous fields. A recent LLM based model from Waymo [3] demonstrated the benefits of LLMs and Chain-of-Thought (CoT) reasoning, showcasing superior driving performance.

However, many of these applications often require to operate on specialized, domain-specific sensor data. Autonomous Driving (AD), for example, benefits from the use of LiDAR, Radar, GPS, and other modalities for robust perception and situational awareness [4–6]. However, because the current generation of general-purpose LLMs are trained only on corpora of text and naturally occurring images from the internet, they lack the understanding necessary to comprehend domain-specific sensor data.

This effect is illustrated in Figure 1, where we prompt the state-of-the-art OpenAI o1 and Gemini 1.5 Pro models to interpret LiDAR and Radar data to identify any cars in the scene. Both models fail, identifying cars in a scene with no cars in the vicinity. In contrast, a human expert would easily interpret the radar signatures to understand that there were no cars. This example highlights the shallow domain expertise of LLM models in comprehending domain-specific sensor data.

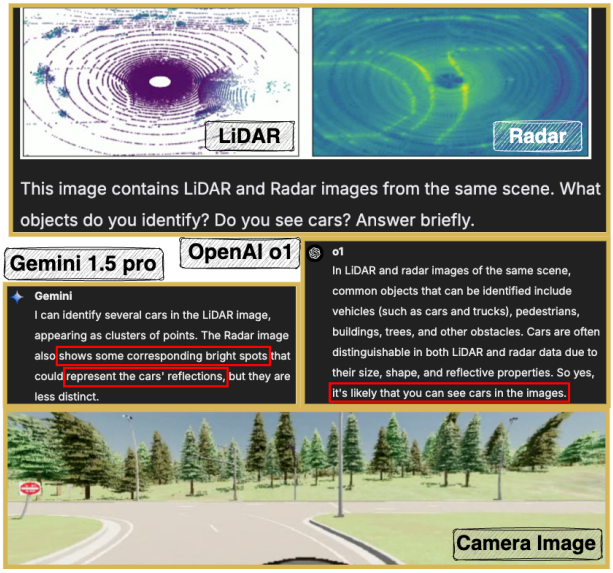


Figure 1: LLMs failing to interpret LiDAR/Radar data

This limitation significantly constrains the applicability of LLMs to tasks that go beyond simple text and camera image prompting. Current approaches to augment LLMs include model retraining, fine-tuning, and adapter layers [7], but these approaches are expensive, time-consuming, and result in an LLM model which is deeply coupled to the application, making them inflexible and non-extensible. Moreover, these techniques require access to the weights of the LLM model, constraining them to use open source models.

SenSys '25, May 6–9, 2025, Irvine, CA, USA

ACM ISBN 979-8-4007-1479-5/25/05.
https://doi.org/10.1145/3715014.3724379

In this project, the proposed framework as shown in Figure 2 offers a modular platform for sensor fusion. This platform enables the use of LLMs in applications that require complex domain knowledge and operate on domain-specific sensor data. Our architecture cleanly decouples the sensor data processing and application-specific domain expertise from the generalist reasoning processes of LLMs. We achieve this by employing independent specialist models to synthesize domain-specific information into comprehensible findings in the form of human-interpretable text. These sensor insights are fused together in the knowledge fusion stage before being passed to the Generalist LLM for decision-making, enabling downstream applications.

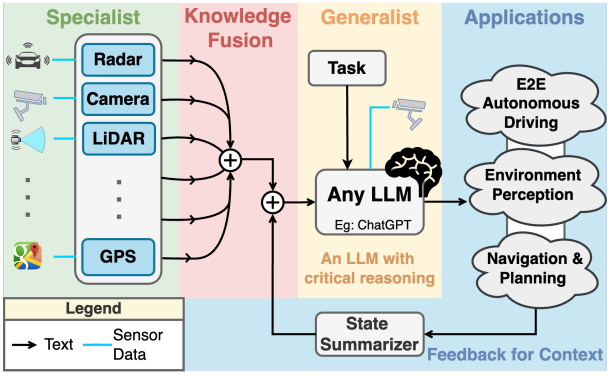


Figure 2: Plug-N-Play Multi-Modal Sensor Fusion Platform

The focus of this platform is for End-to-End AD and the CARLA simulator [8] provides an environment to simulate driving scenarios in real-time allowing for proper evaluation of the model. The simulator also provides sensor outputs such as camera video, radar measurements, and LiDAR point clouds, which serve as inputs to the specialist models.

As for this project, my contributions are is listed as follows:

- **Data Collection:** Collected vast amounts of labeled sensory data (Camera, LiDAR and Radar) in the CARLA simulator which contains frame-by-frame real-time information of all actors in the simulation in a structured format for captions.
- **Captioning Models:** Developed a working proof of concept for LiDAR and Radar captioning models by reducing the problem of caption generation down to classification showing high degree of accuracy.
- **Efficient Pipelines:** I have developed efficient pipelines for large scale data collection and model training on the Nautilus cluster which can further be scaled for training different models.

2 Related Works & Background

2.1 Reasoning in LLMs

LLMs demonstrate remarkable logical reasoning, critical thinking, and decision-making capabilities, and an ability to transfer these skills to new and out-of-distribution scenarios, making them excellent generalists [9–11]. This is largely due to their training on broad and diverse corpora of internet-scale datasets of text and images. Furthermore, recent advancements such as Chain of Thought (CoT) reasoning [12] and self-reflection [13] techniques significantly enhance the quality of reasoning and decision-making abilities of LLMs.

2.2 Adapting LLMs for Sensor Data

Many applications could significantly benefit from the decision-making capabilities of LLMs. However, as alluded to earlier, the use of LLMs often requires access to weights or expensive training. Recent work has attempted to address this issue by feeding raw sensor data directly into an LLM in the form of arrays of structured text [14] or in the form of summarized trends of the data [15]. However, these methods are limited in the complexity and the amount of sensor data they can handle. Moreover, as shown in [16], default tokenizers of LLMs are not well-suited to handle raw data formatted as text. [17] and [18] attempt to better interface and integrate sensor data with an LLM by building a custom tokenizer that addresses the limitations of representing data in the form of text. However, these methods again require access to the weights of the LLM model. [19] introduces a CNN-based classifier to extract activity labels, which are then formatted into structured natural language prompts for LLM inference. However this method relies heavily on pre-classified labels, making it susceptible to errors from upstream models. These papers further solidify the reason to use our platform for sensor fusion via decoupling.

2.3 LLMs in Autonomous Driving

This paper [20] highlights the various AD tasks that benefit from the use of LLMs. In particular, [21] and [22] demonstrate the benefits of LLMs and CoT reasoning for End-to-End (E2E) AD applications. This paper [23] fuses object-level vectorized sensor data with a pre-trained LLM. The sensor data is converted to structured language via vector of numbers and structured language generator. This approach enhances interpretability and is very similar to the proposed platform, but it lacks robust sensing as it only operates on camera data and simulation is performed in open-loop fashion only leading to cumulative errors and inability for real-time adaptation. In contrast, [24] introduces a closed-loop E2E driving model by encoding navigational instructions, camera images, and current state of the vehicle into tokens and directly feeding them to a frozen LLM model. They also have curated a large dataset for AD purposes but they lack in utilizing radar sensor data and do not evaluate their model in diverse scenarios like un-protected left turns, etc.

2.4 Captioning Models

Recent progress in foundation models has substantially advanced scene understanding by bridging diverse sensor modalities with natural language. In autonomous driving, integrating camera and LiDAR data has enabled robust scene captioning, object detection, and multi-task reasoning. Foundation models originally developed for image-language tasks have shown remarkable progress in recent times as indicated by this survey paper [25]. It highlights how large-scale image-text pretraining yields rich semantic representations that are adaptable to complex driving scenarios. These representations can be further enhanced through distillation techniques, as shown in [26] which uses high-capacity 2D backbones to boost LiDAR feature quality for downstream tasks such as captioning.

Traditional LiDAR processing methods, like the PointPillars [27], laid the foundations by encoding sparse point clouds for object detection. A followup paper [28] transfers semantic information from LiDARs by using an SST based transformer architecture to learn corresponding CLIP [29] embeddings via contrastive learning strategy, enabling zero-shot classification, retrieval, and caption generation without requiring extensive LiDAR-text paired datasets. Other methods such as [30] recast 3D scene understanding as a language modeling task by using a view-aware transformer to align sparse LiDAR features with language embeddings.

2.5 Differences with C-Shenron

The past work of Shenron [31] generated realistic Radar data from Camera and LiDAR frames, which was further integrated into CARLA [32] simulator on which I have worked on. This implementation had a crude data collection pipeline and trained an imitation-based model on single GPU. In this project I have updated the pipeline to support large-scale data collection and caption generation using the nautilus cluster, implemented the training pipelines to perform multiple GPU training and optimized the setup for data caching which is essential for reducing the data loading time during model training.

3 Design & Implementation

This section will go over the data collection, caption generation process and the deep learning architectures used for processing sensor data.

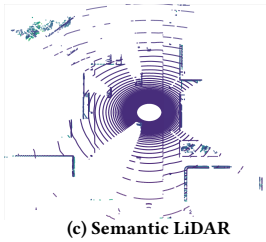
3.1 Data Collection

There are various rule-based expert drivers in CARLA that utilize privileged information from the simulator to imitate human driving behavior. One such expert was used in [4] wherein the agent follows predefined traffic rules, navigates traffic scenarios, and interacts safely with obstacles just like an experienced human driver would perform. This expert also has direct access to detailed map data like lane boundaries, traffic signals, speed limits, and waypoints, which enables precise route planning and rule compliance without relying on raw sensor interpretation. Additionally, the expert bypasses complex object detection, directly retrieving the exact locations, velocities, and classifications of vehicles, pedestrians, and obstacles, thereby eliminating perception errors.

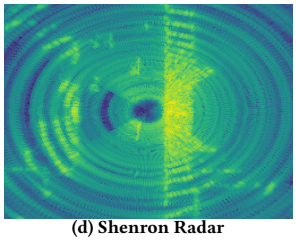
To accelerate data collection, multiple CARLA instances were launched in parallel, each on a different route which helped in generating data across various scenarios and weather conditions. This approach improves the dataset's diversity and richness, and also reducing the collection time from days to hours. Using the Kubernetes cluster, 50 jobs were launched in succession, each corresponding to a distinct CARLA instance for different route-scenario combinations across all 8 CARLA towns (Town01-Town07 and Town10), reserving Town08 and Town09 for evaluation. This results in 70 unique combinations, with each combination repeated thrice, yielding a total of 855k frames over a storage of 4.6TB of data.

(a) Camera Image

(b) Camera Image with Depth Information



(c) Semantic LiDAR

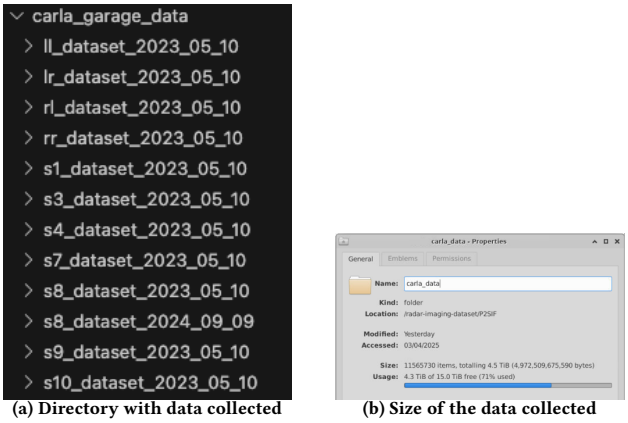


(d) Shenron Radar

Figure 3: Illustrating a sample from the data collected, (a) Camera image of a particular scene at night time, (b) Corresponding depth camera image, (c) Semantic LiDAR point cloud in Birds Eye View perspective and (d) Shenron Radar implemented in CARLA, also in Birds Eye View perspective.

Figure 3 shows a snapshot of the data collected in CARLA from a particular scene during night time. Figure 3a, 3b are the camera and depth camera images collected from First Person View and Figure 3c, 3d is the LiDAR and Radar data collected in Birds Eye View perspective.

Figure 4a shows the directory of the data collected across all scenarios as defined by the CARLA Leaderboard and Figure 4b shows the size of the data collected. For each scenario, the simulation is run across the 8 CARLA towns and for each town there are about an average of 14 routes across which



(a) Directory with data collected (b) Size of the data collected

Figure 4: (a) List of various scenarios for which data was collected in CARLA, (b) Total size of the dataset that has been collected till now.

the data is collected. Sensor data such as Camera, LiDAR and Radar was collected for every route and critical object data such as positions and velocities of objects like vehicles and pedestrians, traffic light status, and traffic signs such as speed limits and stop signs in form of a json file. The latter data will be used to train the captioning models, discussed in below sections.

3.2 Caption Generation

To generating image-caption pair for every frame in the dataset, the real-time information of every actor in the simulation world was recorded along with the sensor data from Camera, LiDAR and Radar, which is stored with time-stamps. Actors consists of entities which are controlled by the CARLA simulator, such as vehicles, motorbikes, pedestrians, traffic lights and traffic signs. The data recorded is formatted into a JSON file with the fields and sub-fields as follows:

(1) For Vehicles:
- ID assigned by CARLA - this is used to identify the ego vehicle among others
- Location in form of (x, y, z) coordinates
- Transform - This contains the pitch, yaw and roll of the vehicle which is used to determine the heading of the vehicle
- Velocity in form of (v_x, v_y, v_z) to validate the heading
- Weather the vehicle is at a traffic light or not
- State of the traffic light if the vehicle is at a traffic light
- Type of the vehicle which is used to distinguish between cars, trucks and bikes

(2) For Pedestrians:
- ID assigned by CARLA
- Location in form of (x, y, z) coordinates
- Transform - Used similarly as above
- Velocity in form of (v_x, v_y, v_z) to validate the heading

(3) For Traffic Signals and Signs:
- ID assigned by CARLA
- Location in form of (x, y, z) coordinates
- Transform - Used similarly as above
- Type - This can be traffic light, stop sign, speed limit and yield signs
- State - This is specific for traffic lights and can be either red, yellow or green
- Description - Contains the speed limit on the sign

To generate captions, the ego vehicle's data was extracted first. Due to the map being large (about 1km x 1km), not all the actors present in the simulator are relevant for the captions, and hence a distance threshold of 40m was used to filter out the relevant actors. To determine the heading of each actor from the pitch and yaw data, the following equations were used:

$$h_x = cos(pitch) * cos(yaw)$$
$$h_y = cos(pitch) * sin(yaw)$$
$$h_z = sin(pitch)$$

where (h_x, h_y, h_z) are the heading vectors. These headings are important to determine if a particular traffic light or traffic symbol applies to the ego vehicle and also to determine the cars that are in and not in the same lane. An example can be seen in Figure 5 which shows the sensor data and the extracted actor locations and headings for actors within a 40m radius which are detectable by LiDAR and Radar.

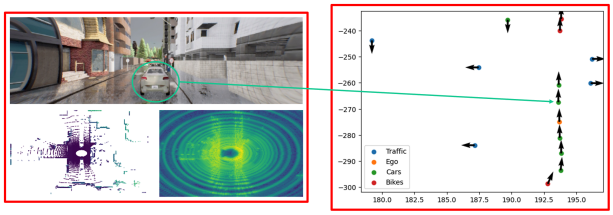


Figure 5: Illustrating a scenario of extracted data from CALRA, left image shows the sensor data of a given scene and right image shows the extracted data from JSON files with the actor locations and individual headings. The green arrow indicates the car in front of the ego vehicle.

For the purposes of this project, I have limited the problem to counting the number of cars and bikes in a given scene and hence define a grammar based caption generation scheme. Going through the collected data, I found that not more than 15 cars and 8 bikes were present in a given scene. Hence to limit the number of possible captions, I have clipped the number of cars and bikes to 10 and 5 respectively, and anything more than that will be detected as "more than 10 cars" and "more than 5 bikes". Therefore, now there are (11 + 1) * (6 + 1), i.e. 84 possible captions for a given scene. Examples of generated captions can be seen in Figure 6.

3.3 Captioning Models

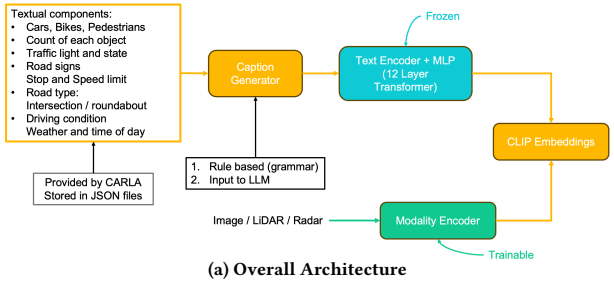
The CLIP [29] architecture introduced by OpenAI, is a foundational model that connects images and natural language

```
"There are eight cars and no bike nearby",
"There are six cars and no bike nearby",
"There are seven cars and no bike nearby",
"No vehicles nearby",
"There is no car and one bike nearby",
```

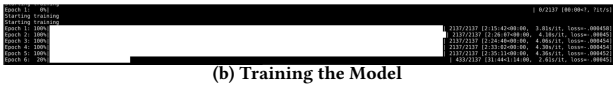
Figure 6: Example captions generated after processing the JSON file.

processing by projecting the learned embeddings down to a common subspace of 512 dimensions. This is done via a contrastive learning technique wherein every textual embedding is enforced to be similar to the correct image and dissimilar to every other image in the batch, and vice-versa for image embeddings. The image encoder uses a Vision Transformer architecture and the textual encoder uses a GPT-2 based transformer model with 12 layers, each layer being 512 units wide, 8 attention heads and totaling about 63 million parameters.

For our setup, the number of captions are fixed and since the textual encoder from CLIP was trained on 200 million image-text pairs from ImageNet, JFT-300M and YFCC datasets, we can utilize this pre-trained encoder for generating the CLIP embeddings. Since we are counting the number of vehicles in the environment, we can use the LiDAR and Radar data to generate captions as they are in birds-eye-view (BEV). We use the same vision transformer as the encoder and pass the LiDAR and Radar BEV images to this encoder. The LiDAR data is obtained as a 3D point cloud image and so we project it down to a 2D plane to generate BEV images. The overall approach is shown in Figure 7a.



(a) Overall Architecture



(b) Training the Model

Figure 7: (a) The overall training pipeline for CLIP based captioning for LiDAR and Radar BEV iamges, (b) Training the CLIP based model takes about 2.5 hours per epoch.

During train time, the image embeddings generated from the encoder are compared with the actual caption embeddings obtained by the text encoder via a cosine similarity loss which is maximized for the correct image-text embeddings. For training the model, I have used 6 A10 GPUs simultaneously on the Nautilus cluster. But one challenge for training on these clusters is that the data and compute nodes have a large physical distance between them, i.e. the data can be in servers on East coast of the US and the GPUs are in Central America. This causes a lot of latency for loading and storing the data on disk, which can be seen as a large training time of 2.5 hours per epoch as seen in Figure 7b. To alleviate the problem of large training time, I am using a caching based mechanism that locally stores the data hashed via user-defined keys. There are a few errors in my implementation due to which the mechanism does not work at the moment. Hence, I have trained the model for 5 epochs only and show the results for the 3 and 5 epoch trained model.

4 Evaluation

Due to high training time as mentioned in Section 3.3, only qualitative evaluation of the model is done. Below are three examples of predicted captions for images from the test-set wherein the left image is the input image and the right image are the captions from ground truth, generated from LiDAR after training for 3 and 5 epochs and radar after training for 5 epochs. Each model predicts the top 3 similar captions according to the cosine similarity between the computed and ground-truth embeddings, which is done to assess the model's versatility is in its predictions. The results are as follows:

- In this example, the actual number of cars and bikes are 5 and 2 respectively. From the Pred 1 generated captions, the radar is spot on in predicting the right number of vehicles whereas the LiDAR predictions are not very accurate even after training from 3 to 5 epochs.

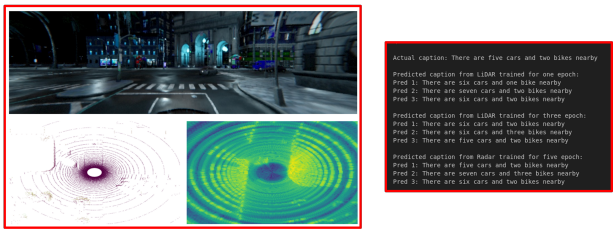


Figure 8: Example of generated caption from test set.

- In this example, there are no vehicles nearby. The radar captioning model again predicts almost correctly the number of vehicles in the scene with an offset of one car. This is mostly because of the surrounding trees in the environment being misclassified as a car which can be surpassed over training multiple epochs. The LiDAR predictions on the other hand are very inaccurate and the predictions get worse over increasing number of trained epochs.
- In this example also there are no vehicles nearby. But the radar captioning predicts one more car than the before example whereas the LiDAR predictions are grossly over estimated.

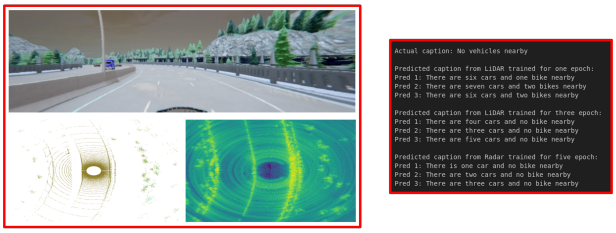


Figure 9: Example of generated caption from test set.

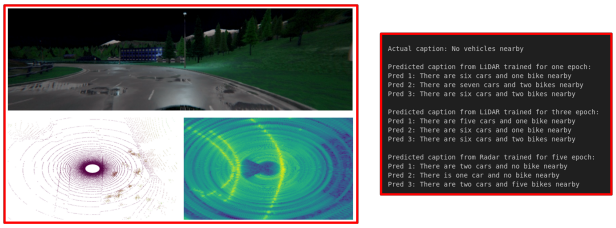


Figure 10: Example of generated caption from test set.

From the above examples, we can conclude that using a ViT based model to process Radar BEV images is very effective in counting the number of vehicles in the scene whereas the LiDAR model is unable to predict any situation correctly and grossly overestimates its output. This is because the LiDAR point clouds are being projected to a BEV image which loses all the rich 3D information of the scene. Hence we should use better architectures such as the sparse spatial transformer from LidarCLIP [33] to retain the dense information from 3D point clouds.

5 Conclusions & Future Work

From the evaluations, we can clearly see qualitatively that the Radar based captioning works well with high degree of accuracy, which can be further improved by training for more epochs. This is a working proof of concept that radar heatmaps (range-angle) can be converted to captions via a deep learning model. In conclusion, I have developed an efficient pipeline for large-scale data collection and model training such that we can now collect terabytes of data overnight and train multiple models simultaneously.

A few challenges faced during this project and corresponding future work are as follows:

- Looking closely at Figure 5, we can see that there is a speed limit sign in the vicinity of the ego vehicle and is also seen in the position plot but the heading of this traffic sign is rotated by 90 degrees. Throughout the data, there are these random 90 degree rotations that has limited my approach for generating only LiDAR and Radar based captions. I propose to fix this issue and work on the camera modality for a finer caption generation instead of counting vehicles.
- The data caching implementation is not able to retrieve the data from keys due to an internal error, which I propose to fix to reduce training time drastically. This will also enable metric based evaluation for the model to provide better insights into model performance.
- Feeding LiDAR BEV images does not yield good results and so I propose to use the SST Trasnformer architecture from LidarCLIP [33] paper to improve the accuracy of the model.
- Due to shortage of time and lots of issues with setting up the pipelines, the project timeline was delayed and hence after training these captioning models, I propose to implement LLM based driving models that will predict waypoints and future driving actions for safe autonomous driving.

References

[1] Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning, 2023.

[2] J. Huang and K. Chang. Towards reasoning in large language models: A survey, 2023.

[3] J. Hwang and et. al. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024.

[4] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger. TransFuser: Imitation With Transformer-Based Sensor Fusion for Autonomous Driving . *IEEE TPAMI*, 45(11):12878–12895, November 2023.

[5] Kshitiz Bansal, Keshav Rungta, and Dinesh Bharadia. Radsegnet: A reliable approach to radar camera fusion, 2022.

[6] Y. Chae, H. Kim, and K. Yoon. Towards robust 3d object detection with lidar and 4d radar fusion in various weather conditions. In *CVPR 2024*.

[7] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[8] A. Dosovitskiy and et. al. CARLA: An open urban driving simulator. PMLR'2017.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models, 2024.

[11] Andrew Szot, Bogdan Mazoure, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons, 2024.

[12] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[13] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Sijie Ji, Xinzhe Zheng, and Chenshu Wu. Hargpt: Are llms zero-shot human activity recognizers? *arXiv preprint arXiv:2403.02727*, 2024.

[15] Xiaomin Ouyang and Mani Srivastava. Llmsense: Harnessing llms for high-level reasoning over spatiotemporal sensor traces. *arXiv preprint arXiv:2403.19857*, 2024.

[16] Dimitris Spathis and Fahim Kawsar. The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language

models. *arXiv preprint arXiv:2309.06236*, 2023.

[17] Zechen Li, Shohreh Deldari, Linyao Chen, Hao Xue, and Flora D Salim. Sensorllm: Aligning large language models with motion sensors for human activity recognition. *arXiv preprint arXiv:2410.10624*, 2024.

[18] Che Liu, Zhongwei Wan, Sibo Cheng, Mi Zhang, and Rossella Arcucci. Etp: Learning transferable ecg representations via ecg-text pre-training. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8230–8234. IEEE, 2024.

[19] Kevin Post, Reo Kuchida, Mayowa Olapade, Zhigang Yin, Petteri Nurmi, and Huber Flores. Contextllm: Meaningful context reasoning from multi-sensor and multi-device data using llms. In *Proceedings of the 26th International Workshop on Mobile Computing Systems and Applications*, HotMobile '25, page 13–18, New York, NY, USA, 2025. Association for Computing Machinery.

[20] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. A survey of large language models for autonomous driving. *arXiv preprint arXiv:2311.01043*, 2023.

[21] Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. Driving with llms: Fusing object-level vector modality for explainable autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14093–14100. IEEE, 2024.

[22] Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, et al. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024.

[23] Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. Driving with llms: Fusing object-level vector modality for explainable autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14093–14100, 2024.

[24] Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L. Waslander, Yu Liu, and Hongsheng Li. Lmdrive: Closed-loop end-to-end driving with large language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15120–15130, 2024.

[25] Sheng Luo, Wei Chen, Wanxin Tian, Rui Liu, Luanxuan Hou, Xiubao Zhang, Haifeng Shen, Ruiqi Wu, Shuyi Geng, Yi Zhou, Ling Shao, Yi Yang, Bojun Gao, Qun Li, and Guobin Wu. Delving into multimodal multi-task foundation models for road scene understanding: From learning paradigm perspectives. *IEEE Transactions on Intelligent Vehicles*, pages 1–25, 2024.

[26] Gilles Puy, Spyros Gidaris, Alexandre Boulch, Oriane Simeoni, Corentin Sautier, Patrick Perez, Andrei Bursuc, and Renaud Marlet. Three Pillars Improving Vision Foundation Model Distillation for Lidar . In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21519–21529, Los Alamitos, CA, USA, June 2024. IEEE Computer Society.

[27] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2019.

[28] Georg Hess, Adam Tonderski, Christoffer Petersson, Kalle Åström, and Lennart Svensson. Lidarclip or: How i learned to talk to point clouds, 2023.

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[30] Senqiao Yang, Jiaming Liu, Ray Zhang, Mingjie Pan, Zoey Guo, Xiaoqi Li, Zehui Chen, Peng Gao, Yandong Guo, and Shanghang Zhang. Lidar-llm: Exploring the potential of large language models for 3d lidar understanding, 2023.

[31] Kshitiz Bansal, Gautham Reddy, and Dinesh Bharadia. Shenron - scalable, high fidelity and efficient radar simulation. *IEEE Robotics and Automation Letters*, 9(2):1644–1651, 2024.

[32] Pushkal Mishra, Satyam Srivastava, Jerry Li, Kshitiz Bansal, and Dinesh Bharadia. Demo abstract: C-Shenron- a realistic radar simulation framework for CARLA. *The 23rd ACM Conference on Embedded Networked Sensor Systems (SenSys '25)*, 2025.

[33] Georg Hess, Adam Tonderski, and Kalle Petersson, Christoffer. Lidarclip or: How i learned to talk to point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.