

# From Vision to Understanding: Leveraging VLMs to enable autonomous driving decisions.

- **Group Members**

- Sridharan Subramanian: ss233
- Brijesh Muthumanickam: brijesh2
- Murali Kartheek Gandiboyina: mkg7
- Sai Rohit Muralikrishnan: srm17

- **Project definition and goals**

Our project focuses on integrating Vision Language Models (VLMs) with autonomous driving systems to enhance their decision-making capabilities and improve overall driving performance. VLMs, which are trained on large-scale web data, combine visual and language understanding, making them powerful tools for interpreting complex driving environments. By using these models, we aim to introduce advanced reasoning into autonomous systems, enabling them to make more informed and generalizable driving decisions.

The core idea behind our approach, called Drive Language Model, is to take visual input from multiple camera views and use the model's reasoning ability to answer critical questions about driving situations, such as traffic conditions, potential hazards, or navigation choices. The ultimate goal is to create an end-to-end driving system that can respond intelligently to complex real-world scenarios, improving both safety and communication with users.

- **Interesting findings.**

One of the most exciting developments in this field is the rise of advanced Vision Language Models (VLMs) like PixMo and MoLMo. These models are capable of understanding and reasoning about both visual and textual inputs, which is pretty impressive. They're trained on huge datasets and can generate text descriptions from images or interpret images based on text. What's really surprising is how they can do things like answer detailed questions about an image, generate captions, or even figure out relationships between objects often in real-time.

For autonomous driving, VLMs like PixMo and MoLMo offer huge potential. By combining their ability to understand visuals with language-based reasoning, they can help the vehicle make sense of complex driving situations.

## **Introduction:**

### **- Problem Motive**

Autonomous driving systems have come a long way, but they still face significant challenges in dealing with the complexity and unpredictability of real-world driving. While current systems can handle many routine situations, they often struggle with more nuanced scenarios, such as interpreting complex road signs, understanding changing traffic conditions, or making quick decisions in unexpected situations. These systems typically rely on pre-programmed rules and sensor inputs, which can limit their flexibility and ability to adapt to dynamic environments. To overcome these challenges, our project aims to integrate Vision Language Models (VLMs) with autonomous driving systems to enhance their decision-making capabilities.

VLMs combine visual understanding with language-based reasoning, allowing the system to interpret both the environment and contextual information in a more human-like way. By processing input from multiple cameras and using the model's reasoning abilities, the system can better understand complex driving situations—such as assessing traffic flow, identifying potential hazards, or evaluating different navigation routes. Our goal is to create an intelligent, end-to-end driving system that not only perceives the world around it but can also reason and make informed decisions in real-time. This integration of vision and language understanding has the potential to improve safety, adaptability, and communication with users, making autonomous vehicles more capable and reliable in the real world.

### **- Discuss background material or related work**

Recent advancements in VLMs, such as PixMo, MoLMo, and CLIP (Contrastive Language-Image Pretraining), have significantly improved the ability to link visual and textual information. CLIP, for instance, has shown the power of pairing large-scale image data with natural language to understand both content and context. PixMo and MoLMo take this further by incorporating advanced reasoning, enabling the models to answer complex questions about images or describe scenes in detail. These models have been applied to a variety of tasks, from image captioning to visual question answering, and have demonstrated their ability to understand dynamic and intricate visual environments in a way that was previously out of reach for traditional models.

Related work in this area also includes the application of Transformer-based models, like DETR (DEtection TRansformers), which combine object detection and language models for better contextual understanding. The development of end-to-end deep learning systems for autonomous driving, such as those from Tesla and Waymo, has also contributed valuable insights into how large-scale perception and decision-making models can be deployed in real-world environments. While these systems currently excel at tasks like lane-keeping and obstacle avoidance, the incorporation of VLMs has the potential to push these systems toward

more advanced reasoning, improving their adaptability and robustness in more complex scenarios.

- **Briefly summary.**

*Started with multiple Traditional Approaches which can be potentially combined with VLMS*

*First approach SAM2 and YOLO v8 approach summary*

This project combines segmentation and motion tracking to achieve robust object tracking across video frames. The process begins with YOLOv8, which generates initial object masks in the first frame by identifying regions of interest. These masks are then carried forward to subsequent frames using SAM2, ensuring consistent identification of objects throughout the video without requiring manual input for every frame. To track the motion of these objects, key features are identified within the masked regions and matched across consecutive frames to compute their movement. This information is used to visualize object motion and refine the tracking process dynamically. The integration of segmentation and motion tracking creates an efficient pipeline that eliminates the need for training or extensive user input, making it ideal for applications where accurate and adaptable object tracking is essential.

*Second approach summary: Optical Flow aided Ego-Vehicle Motion Estimation*

In this method, we use the traditional dense optical flow approaches using Lucas-Kanade and Gunnar-Farneback optical flow to get the motion field of objects of interest. This flow is then leveraged to extract the 3D motion field in world coordinates to understand how the objects move with respect to the ego-vehicle. Finally, our idea is to leverage all the six cameras in the nuScenes data collection setup, use its relative geometry to attain the motion direction of the vehicle so that we achieve ego-vehicle motion estimation with only camera's as the sensor modality.

## **Details of the approach:**

### **Tracking and optical Flow with SAM2 and YOLO v8:**

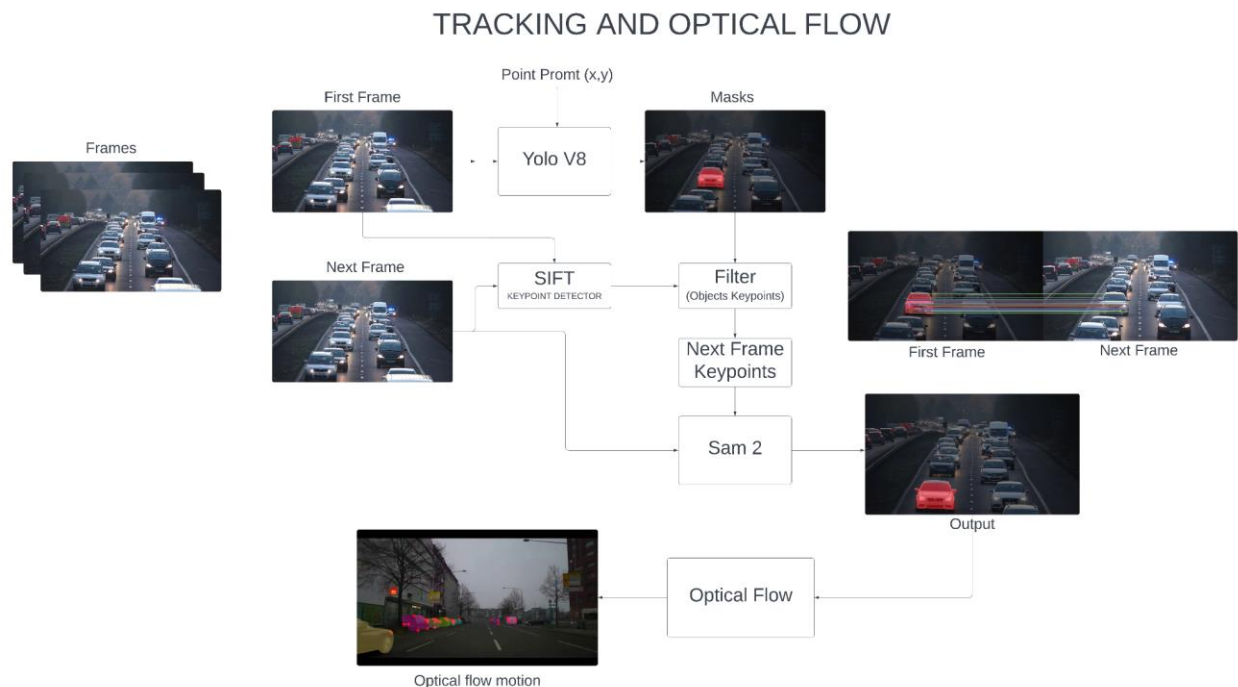
This system is built to track objects across video frames using a combination of segmentation, optical flow, and keypoint tracking. It starts by processing a set of images from a video stored in a directory. The first step is setting up the hardware to make sure the system is using the best available resources, whether that's the CPU, GPU, or MPS on macOS. After the hardware is ready, the system loads two key models: YOLOv8 for detecting objects and generating segmentation masks, and SAM2 for ensuring those masks stay consistent across frames.

Once everything is set up, the system uses YOLOv8 to detect objects and generate masks for the first frame. These masks are then refined and tracked through the video with the help of SAM2, which keeps track of each object and its corresponding mask across frames. If masks are missing in any frame, the system automatically runs YOLO again to regenerate them. This ensures that the system is resilient and can adapt if something goes wrong in the video sequence.

For tracking the movement of the objects, the system uses optical flow. In the multiframe function, keypoints are identified in the object masks using `cv2.goodFeaturesToTrack`. These keypoints represent distinctive points on the objects that can be followed across frames. The system then uses `cv2.calcOpticalFlowPyrLK` to track how these points move from one frame to the next. The motion is visualized by drawing lines to show the trajectory of the keypoints and placing circles at their current positions. This creates a clear visual of how the objects are moving over time.

Finally, the system generates the output by combining the keypoint tracks and the object masks into a single frame, with the motion highlighted over the segmented objects. These processed frames are saved in a results/ folder, so they can be reviewed or analyzed later. The system is designed to be flexible, handling situations where no masks are initially detected by re-running YOLO to regenerate them. This ensures smooth and continuous tracking throughout the video.

Flowchart:



Calculation of depth map:

To calculate depth maps, we used a pre-trained **LiheYoung/depth-anything-small-hf** model using the **transformers** pipeline. Input images were converted to RGB and processed to extract raw depth information. The depth values were normalized, scaled to grayscale (0–255), and saved as **.png** files. Depth arrays from all images were combined into a NumPy array for analysis.

The formula to convert a color intensity **I** to depth **D** in meters is:

$$D = \text{[redacted]} \times \text{Max Depth}$$

Results : please follow the link [output\\_video.mp4](#)

(<https://drive.google.com/file/d/1AuCRCeQvKOmUZx7u1myP4moOfhJruIRZ/view?usp=sharing>)



The red dots are optical flow vectors.

## Spatial Bot:

### Calculating Ego Vehicle's Direction and Motion Using Optical Flow

Calculating the ego vehicle's direction and motion using optical flow involves leveraging motion vectors from consecutive frames to estimate movement relative to the road. This approach captures vehicle motion by analyzing pixel-level displacements, focusing on the road region. The methodology was demonstrated in our earlier work.

We aimed to enhance this process by using a Vision-Language Model (VLM) to understand the scene, detect cars, and determine their directions. Based on the directions of the cars detected by the VLM, we intended to further validate and refine the ego vehicle's motion and direction, thereby supporting the work already accomplished.

### Enhancing Scene Understanding with Vision-Language Models

To support and enhance optical flow-based motion estimation, we integrated a Vision-Language Model (VLM) to:

1. Understand the scene present in the camera's field of view.
2. Identify the movement direction of other vehicles in the scene.
3. Cross-reference optical flow data with the directional insights from SpatialBot to validate and refine motion estimates.

This additional analysis ensures robustness, especially in dynamic traffic scenarios.

### Why SpatialBot as the Vision-Language Model?

We employed SpatialBot, a compact and efficient VLM with 3 billion parameters, for its exceptional spatial reasoning capabilities. Key features include:

1. RGB-D Processing: Combines RGB images and depth maps for precise spatial comprehension, aligning depth and RGB inputs to deduce proximity and spatial relationships.
2. Specialized Training: Trained on datasets like SpatialQA and SpatialQA-E, focusing on tasks such as depth estimation and spatial reasoning. Example queries include "What is the direction of movement of a vehicle?"
3. Efficient Depth Handling: Custom depth encoding ensures consistent performance across varied scenarios, critical for autonomous navigation.
4. Proven Accuracy: Benchmarks on SpatialBench demonstrate SpatialBot's superiority in depth estimation and spatial relationship reasoning.

### Prompt Setup:

1. prompt = (

2. ""Are there any vehicles in front of the camera? If so, which vehicle is very near to the camera and in which direction is it moving?""

3. text = (

4. "A chat between a curious user and an artificial intelligence assistant. "


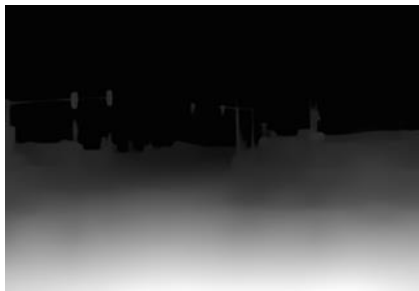

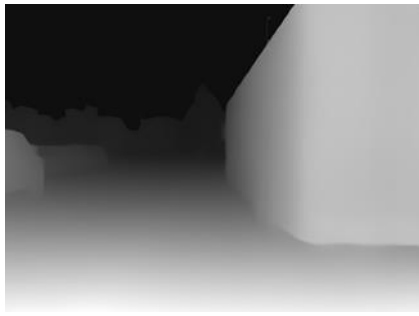
5. "The assistant gives helpful, detailed, and polite answers to the user's questions.\n"



6. "USER: <image 1>\n<image 2>\n"

7. f"{prompt}\n"

8. "ASSISTANT:")

### Output Examples:

RGB Image	Depth Image	SpatialBot
		No, there are no vehicles in front of the camera
		Yes, there is a large blue bus very near to the camera. It is moving towards the right side of the image.

		<p>Yes, there are vehicles in front of the camera. The silver car is very near to the camera and is moving towards the right side of the image</p>
---	--	--

These examples showcase how SpatialBot delivers scene-specific insights, enhancing the reliability of ego vehicle motion estimation and directional validation.

### Exploring Multi-Camera Scenarios:

As demonstrated, SpatialBot performs well in predicting vehicle directions and understanding the scene from a single camera view. However, real-world applications often involve multiple camera views (e.g., six-camera setups). To address this, we aim to evaluate SpatialBot's effectiveness in multi-camera scenarios to ensure it can maintain accurate spatial reasoning and direction prediction across diverse perspectives and fields of view.

### Composite Image Creation for SpatialBot Alignment

SpatialBot's architecture is designed to process a single input image, necessitating a novel approach to integrate information from six camera views in autonomous driving datasets. To align with this architecture while minimizing computational complexity, we developed a method to combine the six camera views into a single composite image while retaining critical spatial and orientation information.

Each camera view was resized to a standardized resolution of **896×448 pixels** to ensure consistency. Depth maps were generated for every view using a pre-trained depth estimation model and resized to match the RGB dimensions. To preserve orientation context, descriptive labels indicating the camera perspective (e.g., front, rear, left) were embedded into both the RGB and depth images. This was achieved by adding semi-transparent overlays and annotating the labels for improved readability.

Finally, the processed views, including both labeled RGB and depth images, were concatenated horizontally to create a unified composite image. This approach allows for the integration of multi-view spatial information into a single input while reducing computational overhead, ensuring compatibility with SpatialBot's architecture and facilitating effective spatial understanding.

Prompt Setup:



```

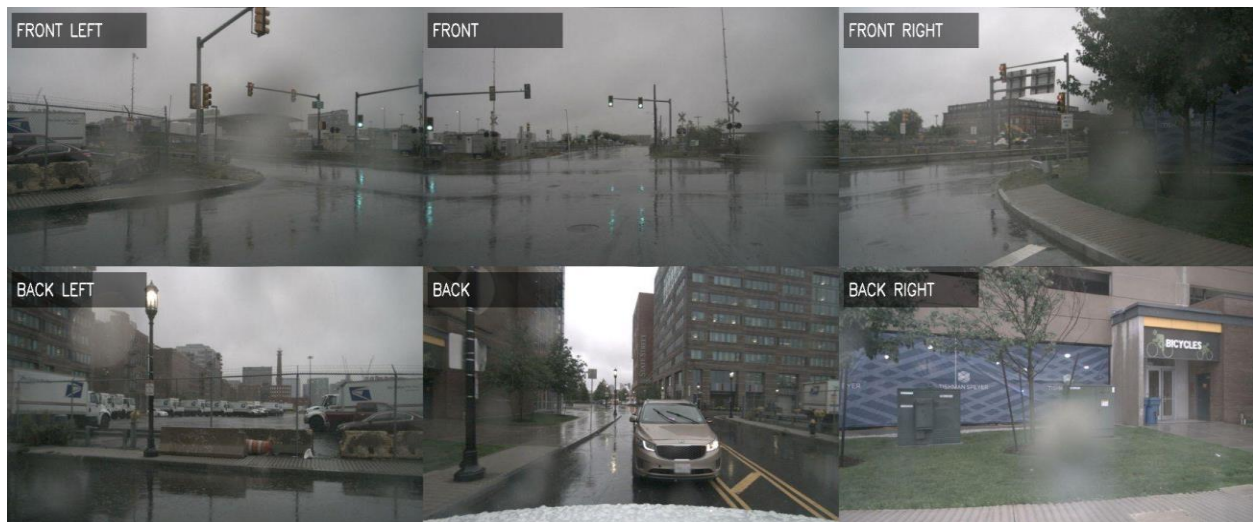
prompt = ("""
Analyze the provided multi-camera grid image. Each camera view is labeled as Front Left, Front, Front Right, Back Left,
Back, and Back Right. Perform the following tasks:
For the vehicle that is nearest to any of the camera, specify:
The camera view(s) in which the vehicle is visible.
The direction the vehicle is moving.""")

text = ( "A chat between a curious user and an artificial intelligence assistant. "
        "The assistant gives helpful, detailed, and polite answers to the user's questions.\n"
        "USER: <image 1>\n<image 2>\n"
        f"{prompt}\n"
        "ASSISTANT:")

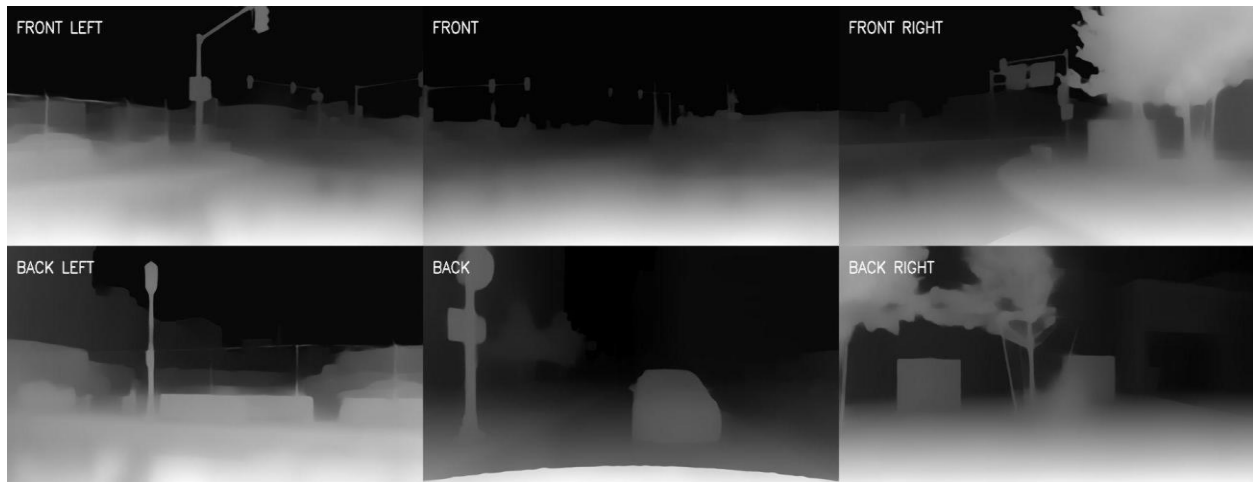
```

Output Examples:

RGB Labelled Composite Image:



Depth Labelled Composite Image:



Result:

**The vehicle nearest to any of the camera views is the car in the middle of the image. It is visible in the Front Left, Front, Front Right, and Front Right camera views. The car is moving towards the right side of the image.**

Despite implementing the composite image methodology for compatibility with SpatialBot's architecture, the output results were not satisfactory. We believe the following factors contributed to the suboptimal performance:

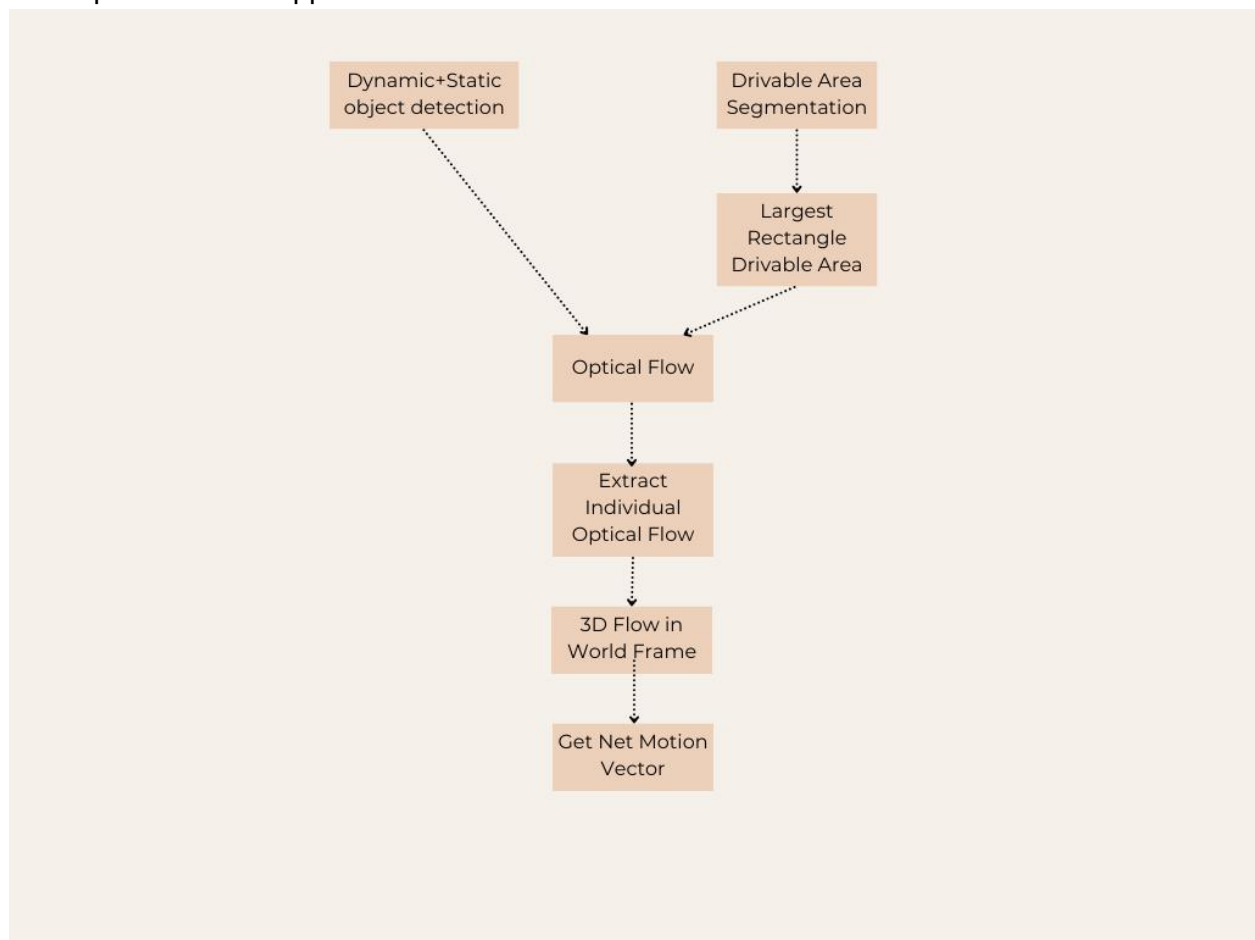
**Loss of Context:** Spatial continuity between views was disrupted during concatenation.

**Resolution Issues:** Resizing to 896×448 pixels may have caused detail loss critical for spatial reasoning.

**Model Adaptation:** SpatialBot was not fine-tuned for composite images, reducing its performance

### **Optical Flow aided Ego-Vehicle Motion Estimation:**

The Pipeline for this approach is as follows:



### Object Detection + Drivable Area Segmentation:

Here, we implemented a custom trained lightweight YOLOv8 segmentation model to get the instances of drivable area along with the dynamic objects. We get the bounding boxes of following objects with more than 0.5 confidence score:

Static Objects of Interests: Traffic Lights, Fire Hydrants, Stop Signs, Parking Meters, Benches.  
Dynamic Objects of Interests: People, Bicycles, Cars, Motorcycles, Trucks, Buses.

Apart from the above objects, we specifically trained the YOLOv8 segmentation model for extracting irregular shaped drivable areas as we can see here. The training was done with the help of TrainYOLO tool for automating the annotation and training pipeline.

### Largest Rectangle in Drivable Area:

This step is necessary to tackle the irregular shape of the roads especially when there are other vehicles on the road. For this task, we have implemented an optimal histogram based approach to get the largest rectangle that is inscribed in this irregular polygon of road mask obtained from the segmentation.

### Optical Flow:

Here, we deploy classical approaches such as the Gunnar-Farneback optical flow method to obtain the dense motion field for the objects in interest. Instead of calling the optical flow instance for every instance of an object, we propose to find optical flow for the entire image and extract only the necessary coordinates. This way, we run this algorithm without the need of a GPU.

Second and major part of this step is about extracting the drivable area's optical flow. For this, we only calculate the flow for the rectangular box obtained from the previous step. We also tackle the case of occasional poor results from road segmentation, i.e, absence of clear drivable area by assuming a rectangular area just before the front camera CAM\_FRONT from the nuScenes dataset. Here is the visualization:

### 3D Flow in World Frame:

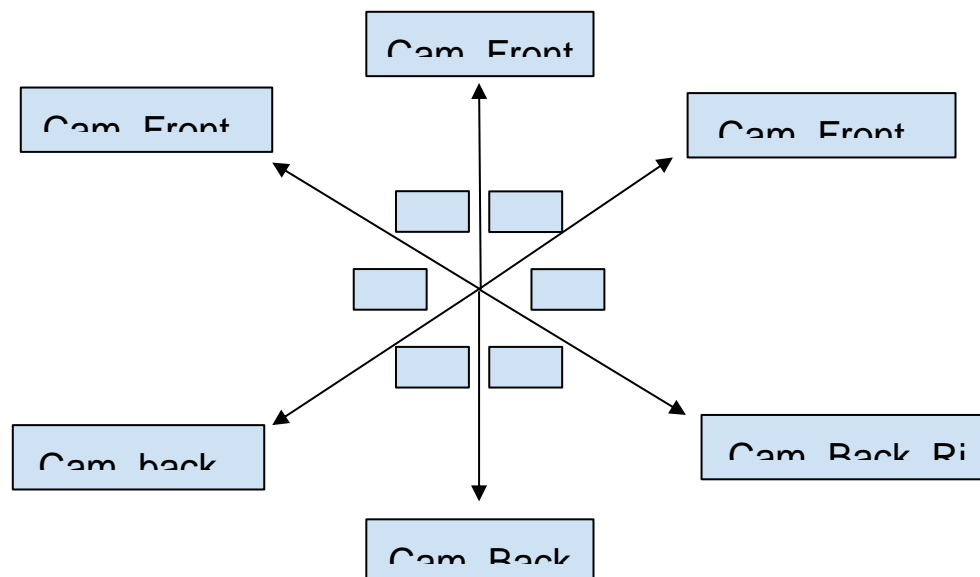
For converting the pixel-wise optical flow into the world coordinate system, we obtained the camera parameters from the nuScenes DevKit. We obtained the camera intrinsics( $K$ ) and the extrinsics( $R, T$ ) to transform the pixel to camera and then in return to world coordinates. Another input is from the optical flow obtained on the drivable area, and finally the depth maps. We calculate depth maps using the Depth Anything Model(DAM) with inference on 100 images from nuScenes. We then utilize all these inputs to perform the transformation from pixel to world coordinate system.

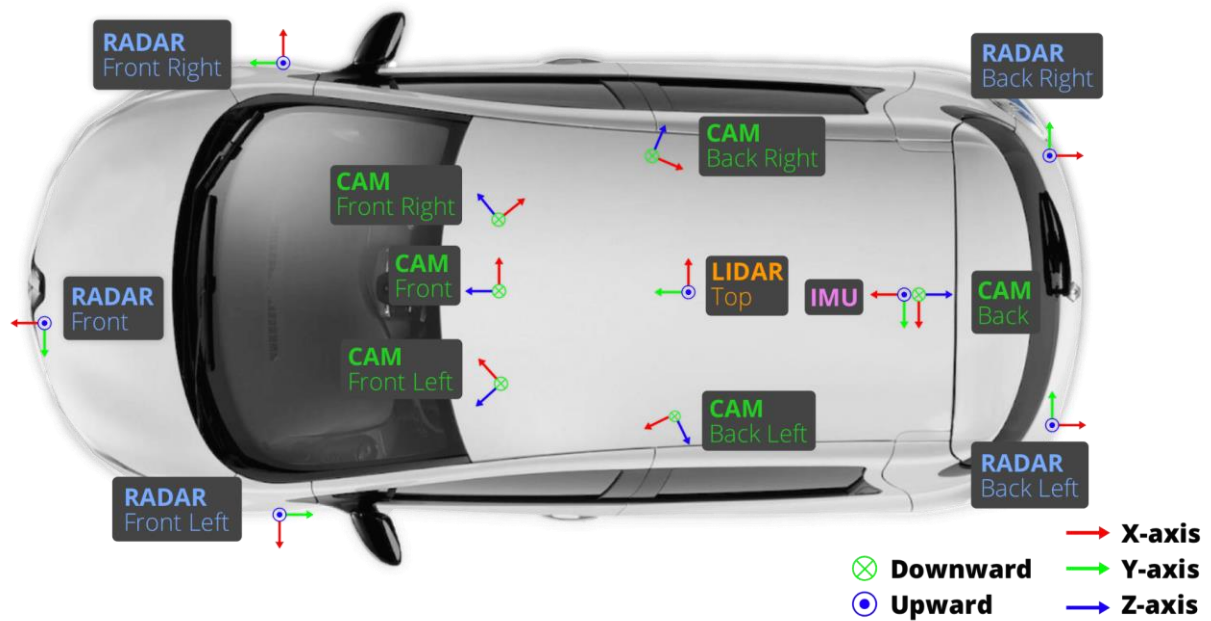
Here, first we project the  $(x, y)$  coordinates to 3D using the camera intrinsic matrix, and also calculate the displaced pixels with the optical flow values. This is projected to 3D as well to finally get the difference between the two 3D coordinates. This is then multiplied with camera extrinsics to obtain them in the world coordinate frame.

#### Net Motion Vector of Ego-vehicle:

In order to get the motion vector of ego-vehicle, we inverse the 3D motion field and ignore the vertical motion. This is visualized as below:

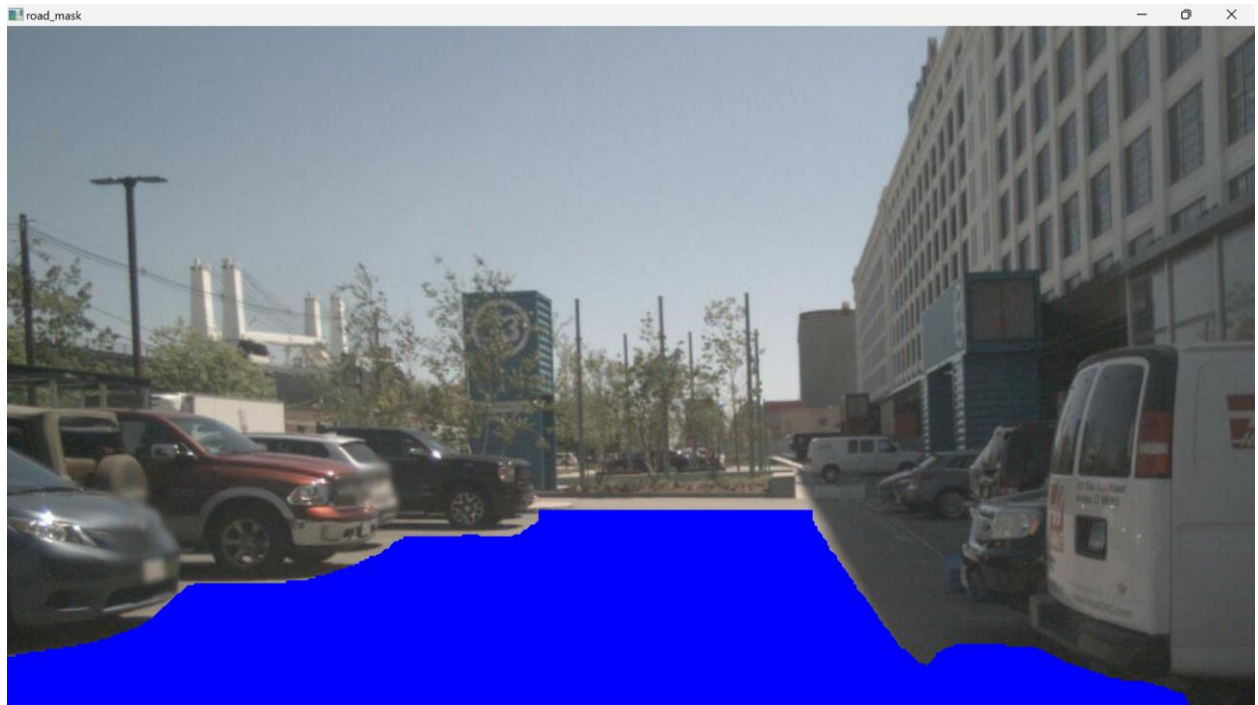
Here, we use the camera setup geometry according to the nuScenes: When we have 3D flow from all the 6 cameras, we use the below geometry to get the component of each motion, and get the net motion direction.



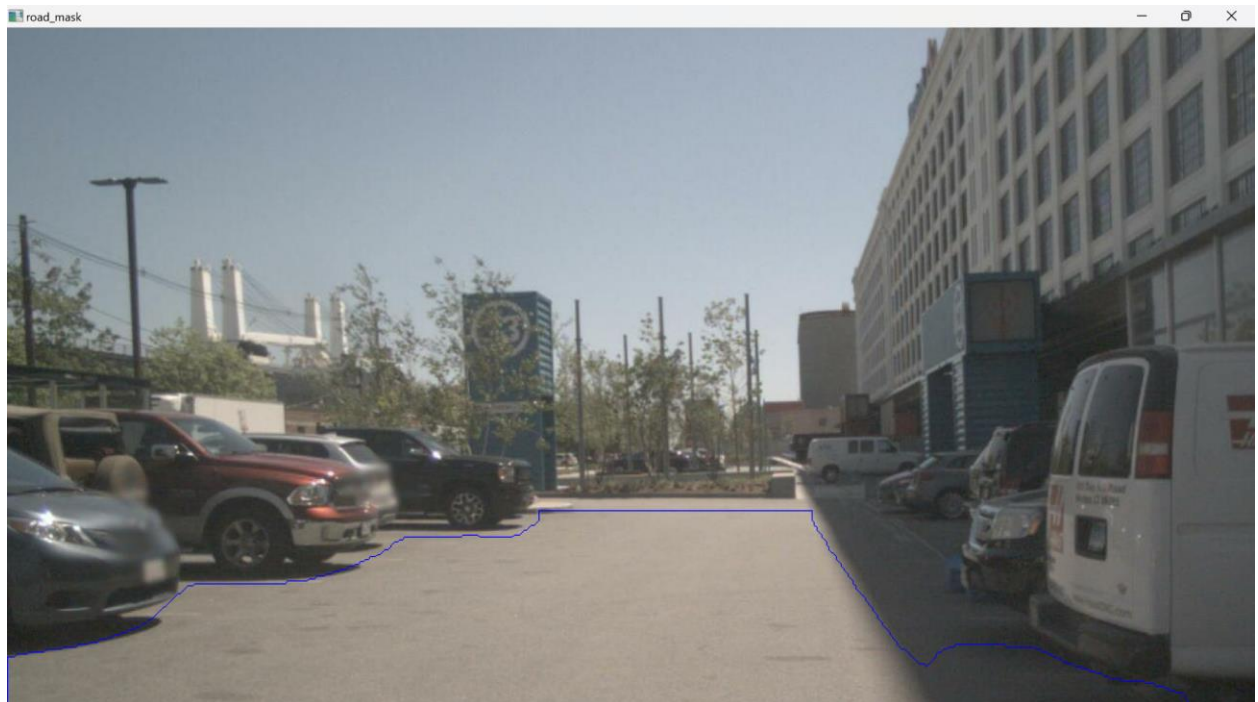


**Results:** Clearly describe your experimental protocols. If you are using training and test data, report the numbers of training and test images. Be sure to include example output figures. Quantitative evaluation is always a big plus (if applicable). If you have example or demo videos, put them on YouTube or some other external repository and include the links in your report.

1. Road Segmentation Mask: Tested under various conditions of shadows and gloomy weather.

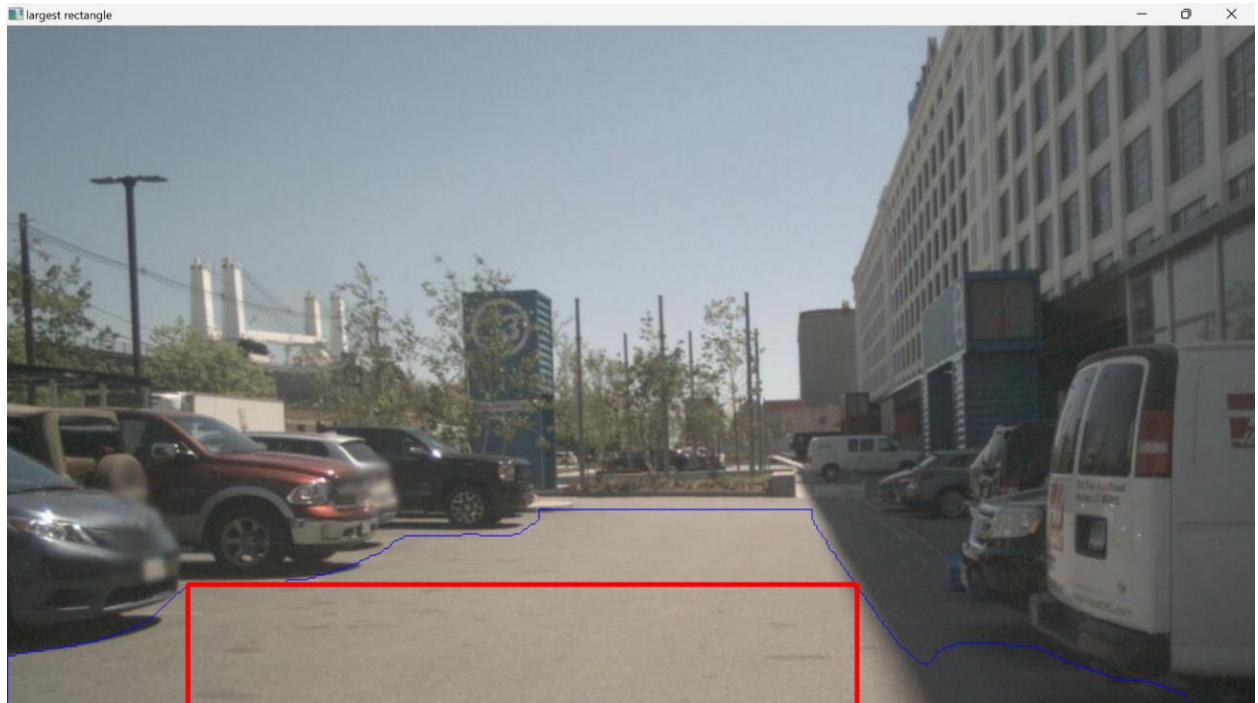


2. Extracted Polygon from Road Mask



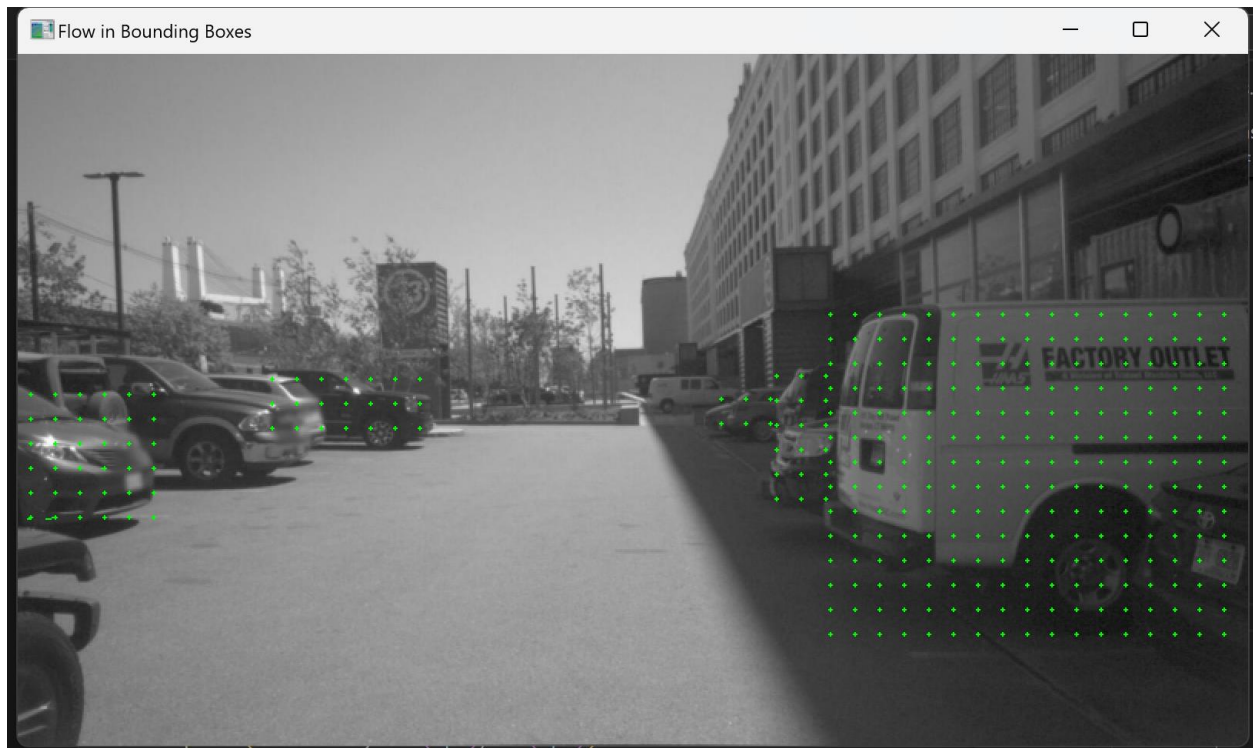


### 3. Largest Rectangle Drivable area - tested in different road shapes and lighting conditions





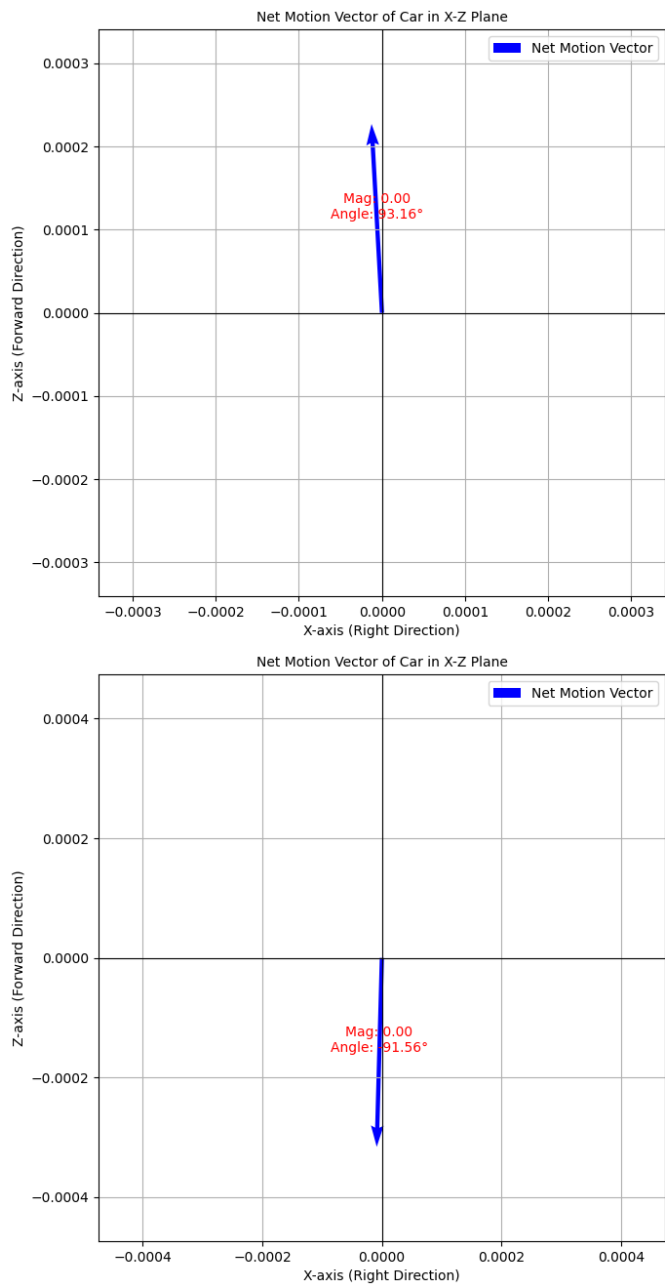
#### 4. Optical flow of dynamic objects:

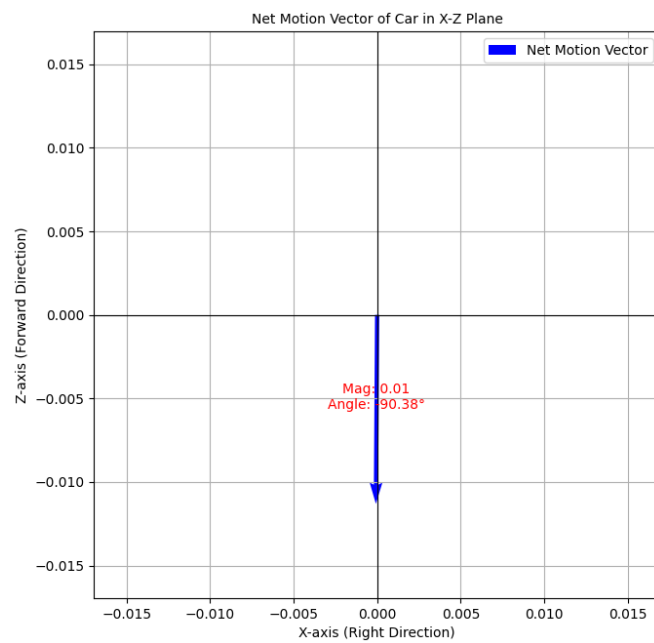
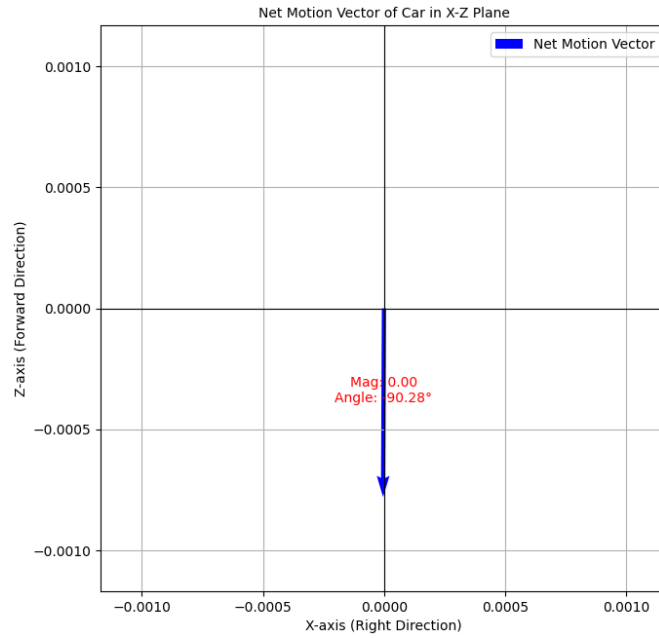


## 5. Optical Flow of drivable area(one of the main static objects in our algorithm)



6. Ego-Vehicle motion direction visualization - This shows 4 consecutive frame's motion direction of the ego-vehicle in 3D world coordinates - estimated from 1 camera using Optical Flow





<https://uofi.box.com/s/rkruwrwfkkg6maa5yb79bfcjezok8bsnp>

**Discussion and conclusions:**

**Tracking and optical Flow with SAM2 and YOLO v8:**

Initially, our approach relied on NanoSAM as a zero-shot learner to generate segmentation masks and facilitate tracking. However, we quickly realized that the processing time required by NanoSAM was a critical bottleneck, making it unsuitable for real-time or near-real-time applications.

To address this limitation, we transitioned to using SAM2 for mask generation. SAM2 demonstrated significant improvements in computational efficiency while maintaining satisfactory accuracy. After obtaining segmentation masks with SAM2, we applied optical flow to track the motion of objects across frames. The integration of SAM2 and optical flow proved to be an effective combination:

**Optical Flow Performance:** The optical flow algorithm performed consistently well in estimating motion between frames. It successfully preserved the continuity of object tracking, even in scenarios with moderate occlusion or variations in object scale and orientation.

**Efficiency Gains:** By leveraging SAM2, we achieved a notable reduction in processing time, making this approach more practical for real-time video analysis.

**Accuracy of Tracking:** Despite improvements in efficiency, the accuracy of segmentation and tracking was slightly dependent on scene complexity. Simple scenes with distinct objects yielded excellent results, whereas cluttered or heavily occluded scenarios introduced minor inaccuracies.

This concludes and discusses the work of: **Optical Flow aided Ego-Vehicle Motion Estimation.**

The idea was to utilize the monocular camera as the primary and only sensor modality. To perform the ego-vehicle pose estimation, there are multiple approaches with extensive learning modules, and this work relies heavily on classical approach. Here, we worked on getting the road segmentation using one of the most lightweight segmentation models to rely as little as possible so we need not segment the entire image. Then to get the largest rectangle inscribed inside the irregular road mask polygon, an histogram algorithm was developed using stack data structure to be as memory and time efficient as possible in case we extend this to real time systems in future. While this paved the way to obtain optical flow from static object(in this case road), we plan to utilize the dynamic objects detected to calculate the ego-vehicle motion field as well. This will involve utilizing a Vision-Language Model (VLM) like DriveLM, CLIP and BLIP VLM architectures or other State-Of-The-Art (SOTA) models to get the accurate direction of the moving objects like cars and pedestrians. This usage of VLM will in addition, serve as an input to the optical flow module as described above in the approach.

Another point of discussion or extension of this work is that: We plan to use all the six cameras of the car in order to work as fail proof to other cameras in use and also to get the complete orientation of the ego-vehicle. This will work by calculating the optical flow in all the camera images, including the dynamic and static objects (like road in this case) and in return utilize the transformation of individual cameras to get the 3D flow in the world coordinate system. Now with the addition of dynamic objects, we anticipate it will work better in the case where road features or other static objects are not prominent and/or simply absent.

Another important detail we noticed is that the scale in 3D flow seems to be showing some discrepancy as it is almost always near zero. The final 3D flow direction is pretty erratic which again depends only on the classical optical flow approach and road features. This can be seen in result image No.6.

### **Connection to course material:**

We implemented optical flow, the classical approach like Lukas-Kanade and Gunnar-Farneback methods. This also involved camera projection matrix calculation and transformation between world and pixel frames - involving topics from MP4, MP5, Lec\_08, Lec\_14, Lec\_15.

### **References**

- <https://github.com/ultralytics/ultralytics>
- <https://docs.ultralytics.com/modes/predict/#key-features-of-predict-mode>
- Sima C., Renz K., Chitta K., Chen L., Zhang H., Xie C., et al., DriveLM: Driving with Graph Visual Question Answering, arXiv.Org, 2023, abs/2312.14150
- Qin T., Li P., Shen S., VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator, IEEE Transactions on Robotics, 2018, 34, 1004–1020
- [https://docs.opencv.org/4.x/da/de9/tutorial\\_py\\_epipolar\\_geometry.html](https://docs.opencv.org/4.x/da/de9/tutorial_py_epipolar_geometry.html)
- <https://github.com/binh234/nanosam>
- Motional. (n.d.). *GitHub - motional/nuplan-devkit: The devkit of the nuPlan dataset*. GitHub. <https://github.com/motional/nuplan-devkit>
- Gopalkrishnan, A., Greer, R., & Trivedi, M. (2024). Multi-Frame, Lightweight & Efficient Vision-Language Models for Question Answering in Autonomous Driving. *arXiv preprint arXiv:2403.19838*.
- <https://github.com/OpenGVLab/InternVL>
- Makiyeh, F., Bastourous, M., Bairouk, A., Xiao, W., Maras, M., Wangb, T. H., ... & Rus, D. (2024). Optical Flow Matters: an Empirical Comparative Study on Fusing Monocular Extracted Modalities for Better Steering. *arXiv preprint arXiv:2409.12716*.
- Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., & Zhao, H. (2024). Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10371-10381).

- Cai, W., Ponomarenko, I., Yuan, J., Li, X., Yang, W., Dong, H., & Zhao, B. (2024). Spatialbot: Precise spatial understanding with vision language models. *arXiv preprint arXiv:2406.13642*.
- Li, J., Li, Z., & Lu, T. (2024). Driving with InternVL: Outstanding Champion in the Track on Driving with Language of the Autonomous Grand Challenge at CVPR 2024. *arXiv preprint arXiv:2412.07247*.
- Gopalkrishnan, A., Greer, R., & Trivedi, M. (2024). Multi-Frame, Lightweight & Efficient Vision-Language Models for Question Answering in Autonomous Driving. *arXiv preprint arXiv:2403.19838*.