End-to-end Autonomous Driving At scale and with Language

Chonghao Sima & Kashyap Chitta

@ CVPR Tutorial

June 18 2024

https://opendrivelab.com

https://cvlibs.net/

OpenDriveLab

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

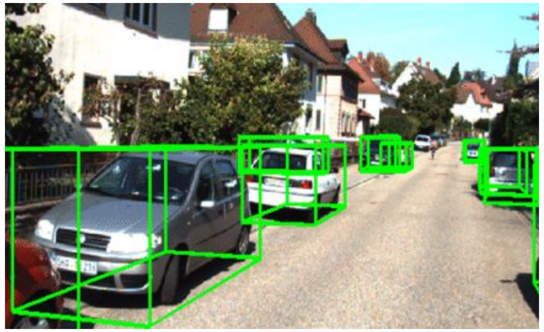
End-to-end Autonomous Driving

Introduction

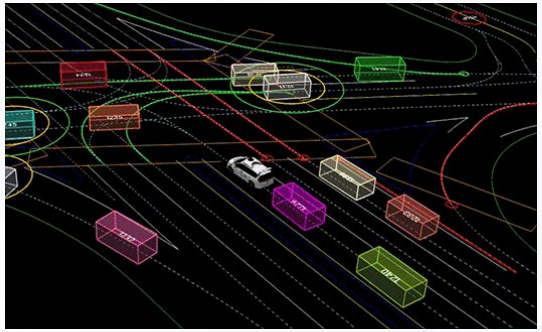
Autonomous Driving (AD) Tasks

Challenge | Various weathers, illuminations, and scenarios

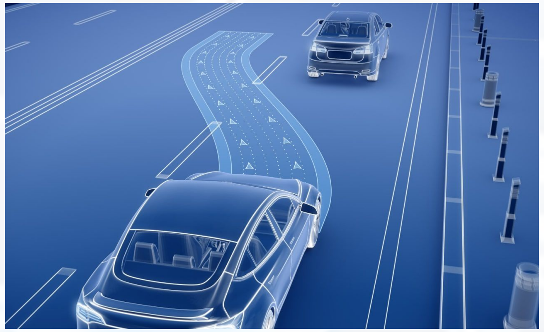
Perception → (Bounding boxes) → Prediction → (Waypoints) → Planning → (Trajectory)



What are around?



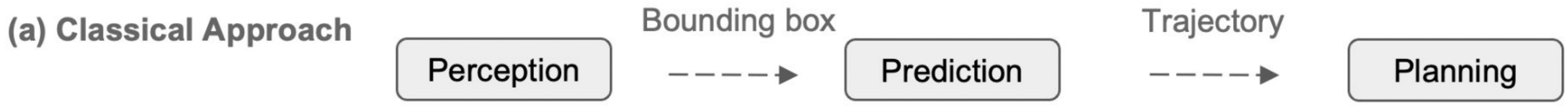
How will they go in the future?



Where should I go?

Conventional Autonomous Driving (AD) Systems

Modular-based



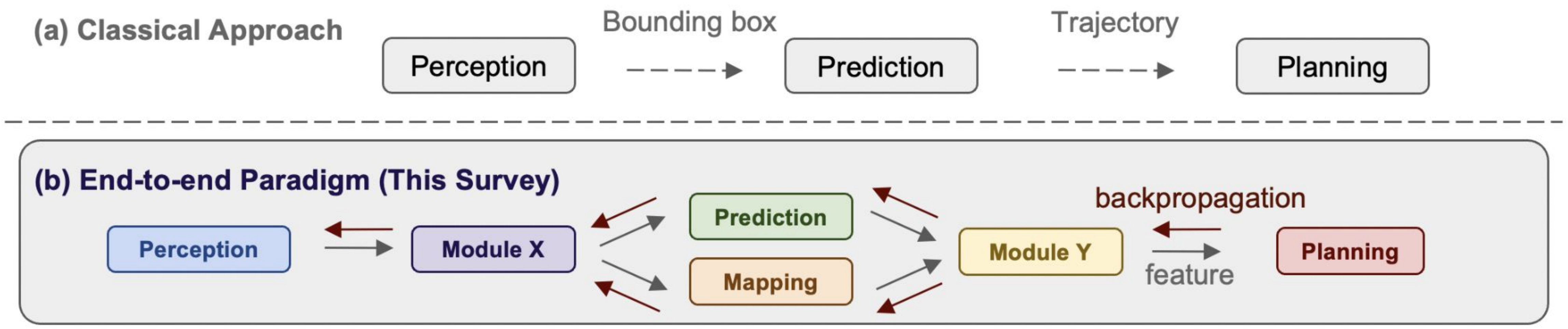
Pros

+ **Independent** teams for module development
  + Dataset friendly
  + Quantitatively evaluation for intermediate tasks
  + Great interpretability
+ Parallel onboard deployment

Cons

- Error accumulation, information loss. **Results, instead of features** are traversed across modules
- Labelling cost
- Computational cost for separate models

Motivation | Why End-to-end (E2E) Autonomous Driving?



End-to-end autonomous driving system - A suite of fully differentiable programs that:

- take raw sensor data as input
- produce a plan and/or low-level control actions as output
- all modules can be optimized via gradient descent

Motivation | Why End-to-end (E2E) Autonomous Driving?

Advantages

- **Simplicity** in combining all modules into a single model that can be **joint trained**
- Preventing cascading errors in modular design
- Directly optimized **toward the ultimate task**, planning / trajectory prediction
- Computational efficiency (all shared backbone), production-level friendly

OpenDriveLab

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

DriveLM:
Driving with Graph Visual Question Answering

https://github.com/OpenDriveLab/DriveLM
https://arxiv.org/abs/2312.14150

Insight | VLM in Robotics / Embodied AI

- How vision-language models trained on Internet-scale data can be incorporated directly into **end-to-end robotic control**
- Goal: to **boost generalization** and enable emergent semantic reasoning

- Robotic tasks naturally fits into language at dissecting tasks step by step using language (prompt)
- Is it the **right way** to open the language tool box as does in Robotics for Autonomous Driving?

Key ingredient(s): huge amount of data (not public) + language prompt to dissect tasks

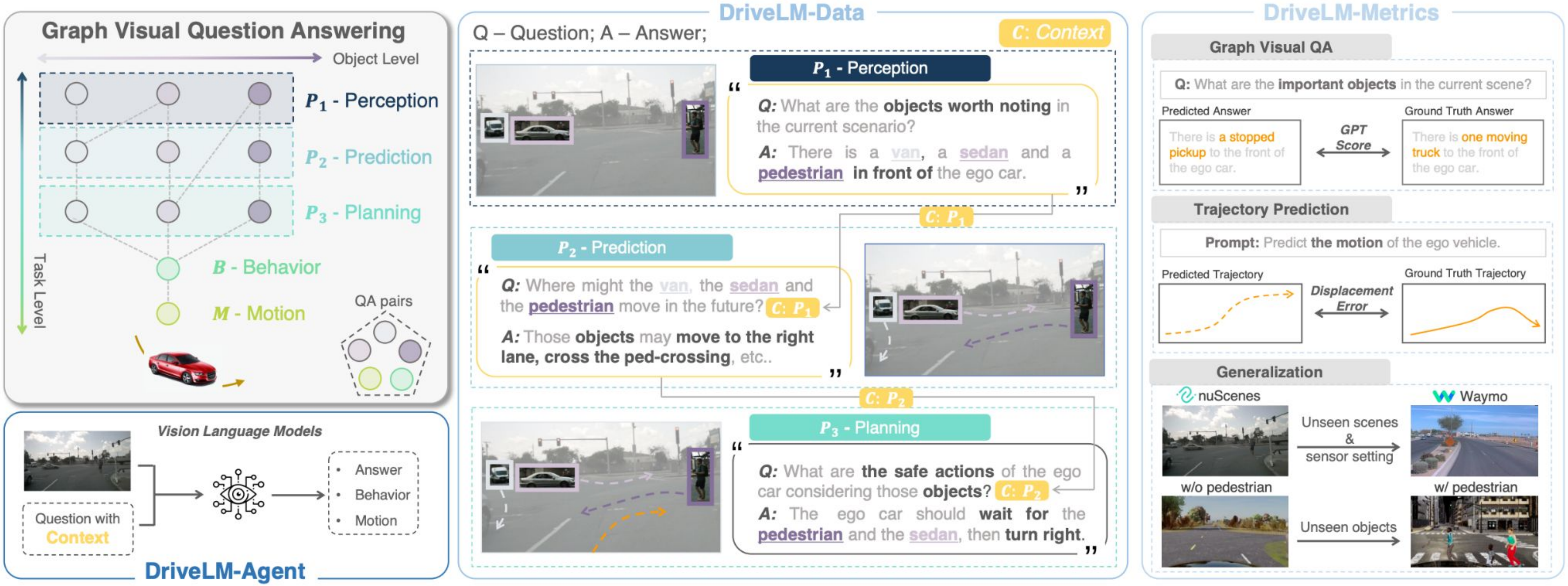
DriveLM | Introduction

- **Generalization** and **Interactivity** in Autonomous Driving
  - Generalized to **unseen** sensor configuration and objects
  - Regional / Global (e.g. European) regulations require **explainability** through interaction

- Recent success in **Vision Language Models**
  - Good **reasoning** ability, enabled by LLM
  - **No BEV** representation, since human do not rely on BEV

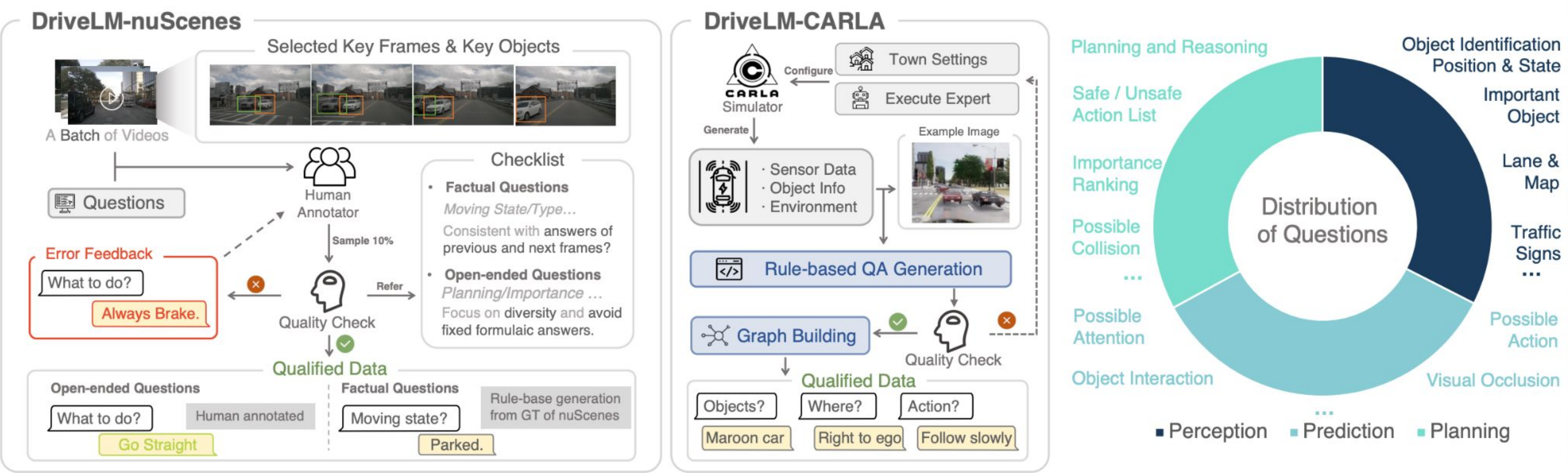
- Why VLM in AD?
  - **Reasoning** ability helps **generalization**
  - **Language** output provide **interactivity**

DriveLM | At a Glance



- The critical part is **Graph Visual QA**, upon which we build **data**, model and metrics accordingly

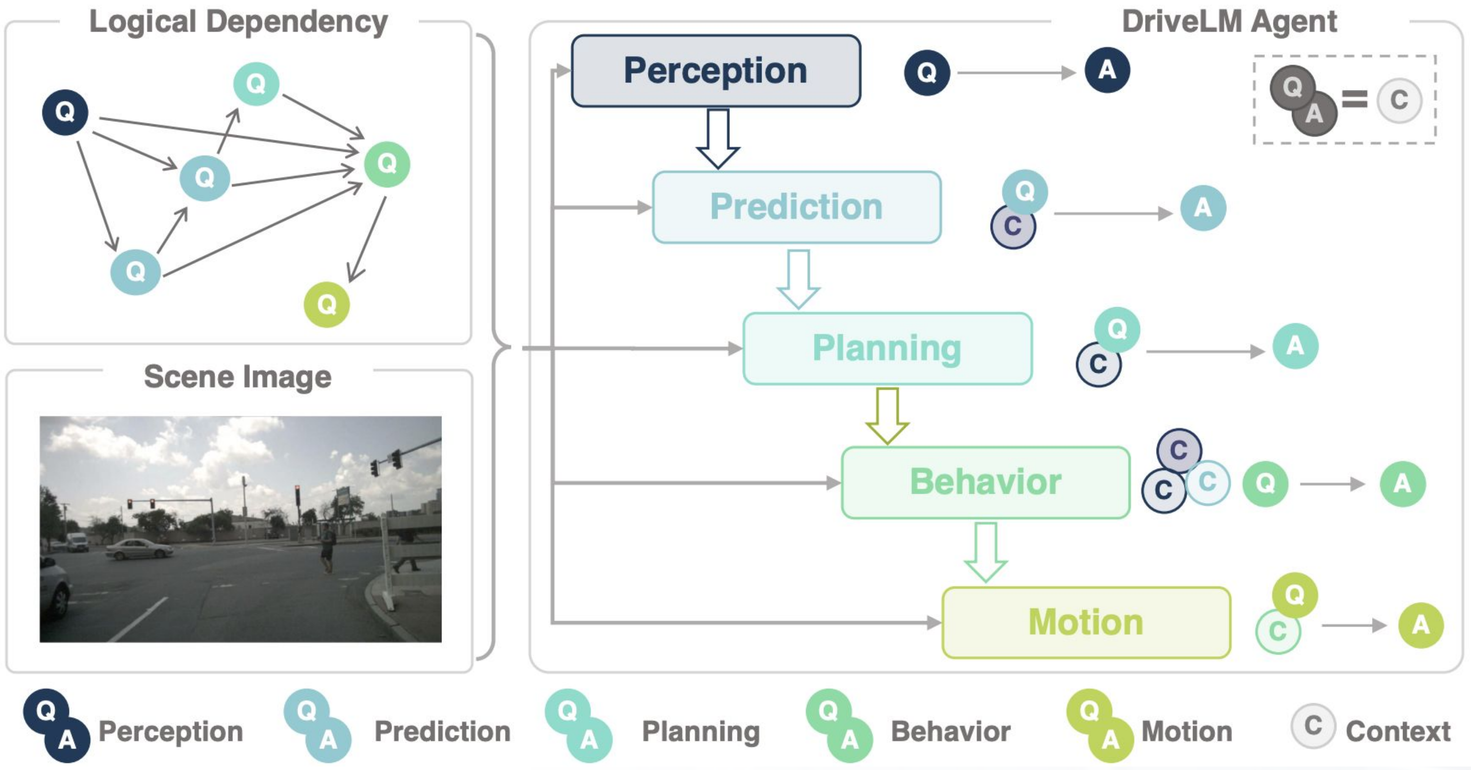
DriveLM | Data



- To ensure the **data quality**, we introduce human annotation with multi-round quality check in nuScenes
- To **scale-up** annotation, we adopt auto-labelling in CARLA

Diversity matters, spanning from perception to prediction and planning

DriveLM | Agent



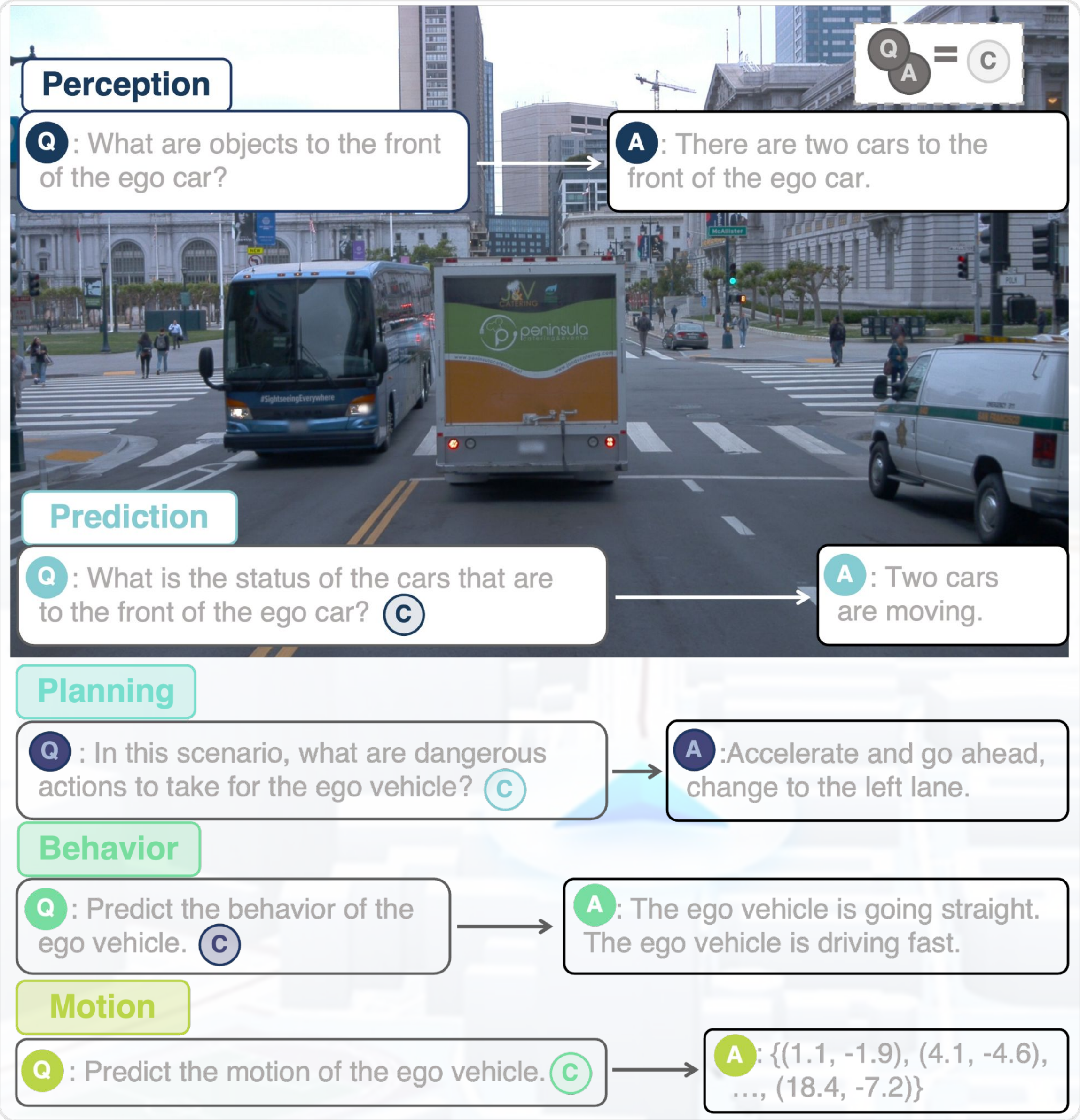
- 🧩 General and scalable VLM architecture
- 🌏 Web-scale pre-training
- 🛠️ Fine-tuned end-to-end for planning
- 💡 Interpretable and interactive

DriveLM - Experiments

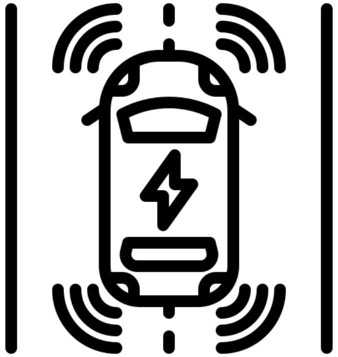
Method	Behavior Context	Motion Context	Behavior (B) Acc. ↑	Behavior (B) Speed ↑	Behavior (B) Steer ↑	Motion (M) ADE ↓	Motion (M) FDE ↓
Command Mean	-	-	-	-	-	7.98	11.41
UniAD-Single	-	-	-	-	-	4.16	9.31
BLIP-RT-2	-	-	-	-	-	2.78	6.47
DriveLM-Agent	None	B	35.70	43.90	65.20	2.76	6.59
	Chain	B	34.62	41.28	64.55	2.85	6.89
	Graph	B	**39.73**	**54.29**	**70.35**	**2.63**	**6.17**

- Trained on DriveLM-Data (nuScenes-based), DriveLM-Agent (ours) gains **better zero-shot** ability on Waymo scenarios, overpassing other methods by a large margin.

- Qualitative result shows that DriveLM-Agent does **understand the unseen scenarios** in some way.

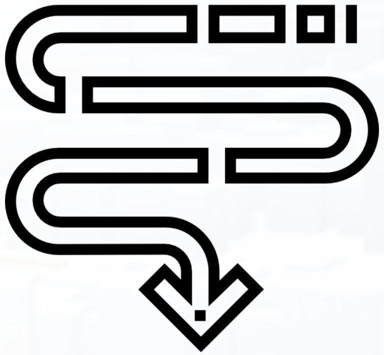


DriveLM - Limitation



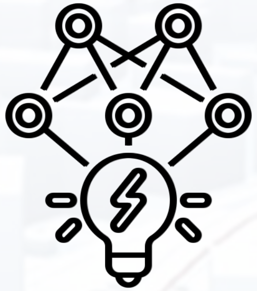
Driving-specific Inputs

DriveLM-Agent cannot handle common setting such as LiDAR or multi-view images as input, limiting its information source.



Closed-loop Planning

DriveLM-Agent is evaluated under an open-loop scheme, while closed-loop planning is necessary to see if it can handle corner cases.



Efficiency Constraints

Inheriting the drawbacks of LLMs, DriveLM-Agent suffers from long inference time, which may impact practical implementation.

One-page Takeaway

- End-to-end Autonomous Driving
  - Challenge: **Generalization & Explainability**
  - Recent trend: use vision language model to **embed "world knowledge"** to solve the challenges.
- DriveLM: Driving with Graph Visual Question Answering
  - Use **Graph VQA** as a proxy task to mimic human's driving logic
  - **Some good result under zero-shot setting, but still far from claiming good generalization.**

OpenDriveLab

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

Driving-with-Language Track
Tutorial of VLM fine-tuning for DriveLM

https://github.com/OpenDriveLab/DriveLM/tree/main/challenge

How to evaluate VQA in driving thoroughly

Q: What are the **important** objects in the current scene?

A: There is a brown SUV to the back of the ego vehicle, a black sedan to the back of the ego vehicle, and a green light to the front of the ego vehicle.

Predict: There is a brown SUV to the back of the ego vehicle.

How to evaluate it and reflect its influence to the following QA?

We want to evaluate the correct part “brown SUV”, and penalise the missing parts “black sedan, green light”, and reflect the effect of missing in the following QA (prediction & planning).

Outline

- Perception
- Prediction
- Planning

Perception

Please Refer to line 29 in evaluation.py

Q: What are the **important** objects in the current scene?

GT: There is a brown SUV to the back of the ego vehicle, a black sedan to the back of the ego vehicle, and a green light to the front of the ego vehicle.

Predict: There is a brown SUV to the back of the ego vehicle.

Input: GT & Predict
Metric：Using BLUE、ROUGE_L and CIDER
Package：language_evaluation.CocoEvaluator

Perception

Please Refer to line 18 in evaluation.py

Q: Are there barriers to the front right of the ego car?

GT: Yes.

Predict: No.

Input: GT & Predict
Metric: Accuracy
Package: sklearn

sum(1 for true, pred in zip(GT, Predict) if true == pred)

Prediction

<c1,CAM_BACK,1088.3,497.5>

Please Refer to line 18 in evaluation.py

Q: What is the future state of <c1,CAM_BACK,1088.3,497.5>?
Please select from A: Turn left、B: Turn right、C: Forward.

GT: A.

Predict: B.

Input: GT & Predict
Metric: Accuracy
Package: sklearn

sum(1 for true, pred in zip(GT, Predict) if true == pred)

Prediction —— Graph evaluation prunes branches.

Please Refer to line 78 in evaluation.py

Q: What object should the ego vehicle notice first when the ego vehicle is getting to the next possible location? What object should the ego vehicle notice second when the ego vehicle is getting to the next possible location? What object should the ego vehicle notice third?

GT: Firstly notice that <c3,CAM_FRONT,1043.2,82.2>. Secondly notice that <c1,CAM_BACK,1088.3,497.5>. Thirdly notice that <c2,CAM_BACK,864.2,468.3>.

Predict: <c3,CAM_FRONT,1043.2,82.2>. <c2,CAM_BACK,864.2,468.3>.

As the perception only predict one important objects. Then we only evaluate that object <c1,CAM_BACK,1088.3,497.5>

Prediction —— Object matching

Please Refer to line 57 in evaluation.py

<c3,CAM_FRONT,1043.2,82.2>

<c2,CAM_BACK,864.2,468.3>

<c1,CAM_BACK,1088.3,497.5>

GT: Firstly notice that <c3,CAM_FRONT,1043.2,82.2>.
Secondly notice that <c1,CAM_BACK,1088.3,497.5>.
Thirdly notice that <c2,CAM_BACK,864.2,468.3>.

Predict: <c1,CAM_FRONT,1040.2,80.2>.
<c2,CAM_BACK,865.2,470.3>.

Then we only keep < c1,CAM_BACK,1088.3,497.5> as GT.
And decrease 2/3 score firstly.

L2(<1088.3, 497.5>, < 1040.2,80.2>) > ∋ ——> Not Match
L2((<1088.3, 497.5>, < 865.2,470.3>) < ∋ ——> Matched!

Len(Matched) / len(GT)

Planning

Please Refer to line 24 in evaluation.py

Q: What actions could the ego vehicle take based on <c2,CAM_BACK,864.2,468.3>?

GT: The action is to keep going at the same speed.

Predict: The action is to keep going.

ChatGPT evaluation

Prompt: We will provide a question and a corresponding two answers. One is ground truth. One is predictions. Assuming the ground truth is 100 score, please score the predict answer. Output the score only.
Q: What actions could the ego vehicle take based on <c2,CAM_BACK,864.2,468.3>?
Ground Truth: The action is to keep going at the same speed
Predict: The action is to keep going.

Planning

Please Refer to line 24 in evaluation.py

<c2,CAM_BACK,864.2,468.3>

Q: What actions taken by the ego vehicle can lead to a collision with <c2,CAM_BACK,864.2,468.3>?

GT: Back up.

Predict: Speed up.

ChatGPT evaluation

Prompt: We will provide a question and a corresponding two answers. One is ground truth. One is predictions. Assuming the ground truth is 100 score, please score the predict answer. Output the score only.
Q: What actions taken by the ego vehicle can lead to a collision with <c2,CAM_BACK,864.2,468.3>?
GT: Back up.
Predict: Speed up.

Planning

Please Refer to line 24 in evaluation.py

Q: In this scenario, what are safe actions to take for the ego vehicle?

GT: Keep going at the same speed, decelerate gradually without braking.

Predict: Keep going at the same speed.

ChatGPT evaluation

Prompt: We will provide a question and a corresponding two answers. One is ground truth. One is predictions. Assuming the ground truth is 100 score, please score the predict answer. Output the score only.
Q: In this scenario, what are safe actions to take for the ego vehicle?
GT: Keep going at the same speed, decelerate gradually without braking.
Predict: Keep going at the same speed.

Evaluation

Please Refer to line 157 in evaluation.py

Final Score = 0.4 * ChatGPT + 0.2 * Language + 0.2 * Match + 0.2 * Accuracy

ChatGPT [0, 100]

Language Score:

1. BLUE [0, 1]
2. ROUGE_L [0, 1]
3. CIDER [0, 10]

Match Score [0, 100]

Accuracy [0, 1]

We weighted and averaged several of the previous scores to get the final score, with ChatGPT Score, Language Score, Match Score and Accuracy having a weight of 0.4, 0.2, 0.2 and 0.2 respectively.

OpenDriveLab

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

THANKS

opendrivelab.com

https://cvlibs.net/

DriveLM

Code + Data

Paper

NAVSIM

Code + Data

Shanghai AI Lab