Scenario-Adaptive Trajectory Refinement

Yang Zhou

Sensetime Research

4 June 2024

自动驾驶之心官网

www.zdjszx.com

Contents

1. Introduction of trajectory prediction.

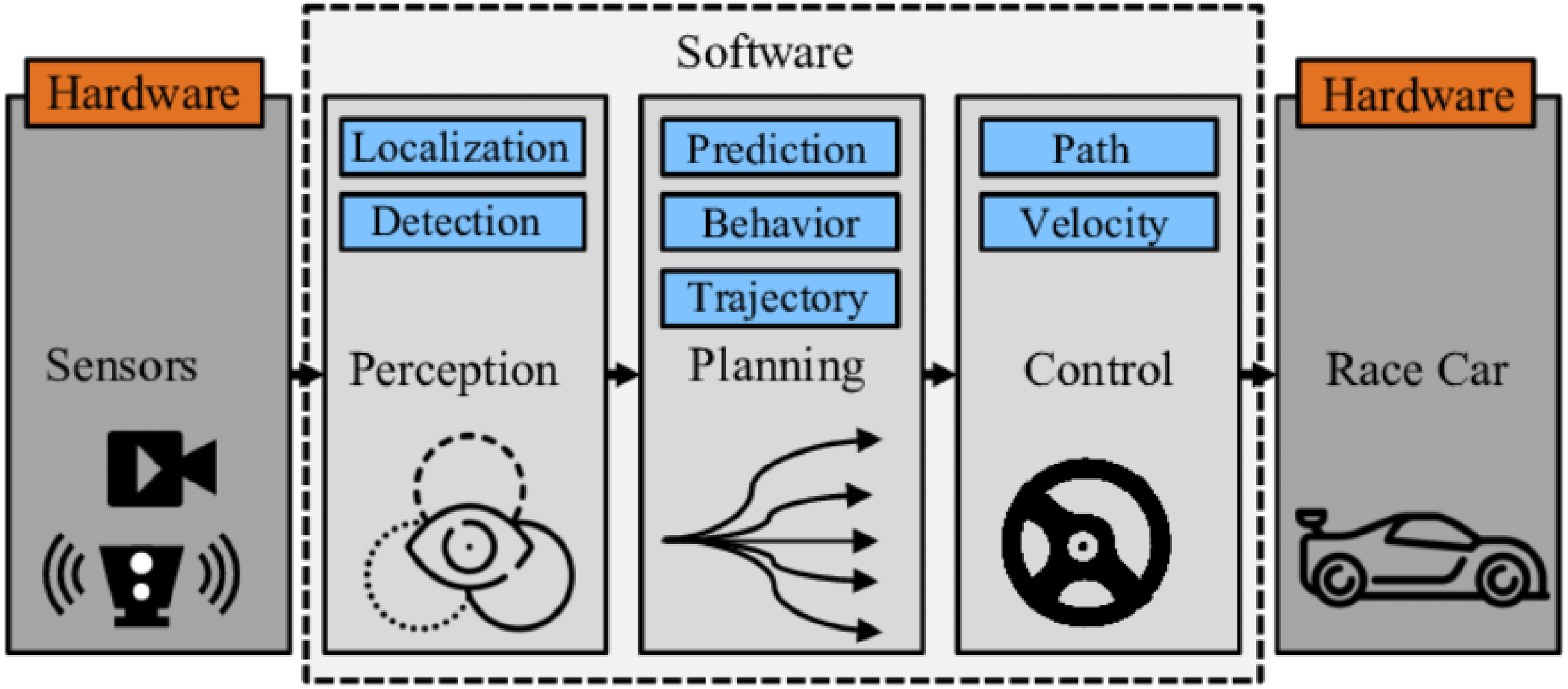
2. What can we learn from human drivers when predcting surounding agents’ future motion?

3. Our SmartRefine Framework.

4. How can refinement benefit the practical use of trajectory prediction methods?

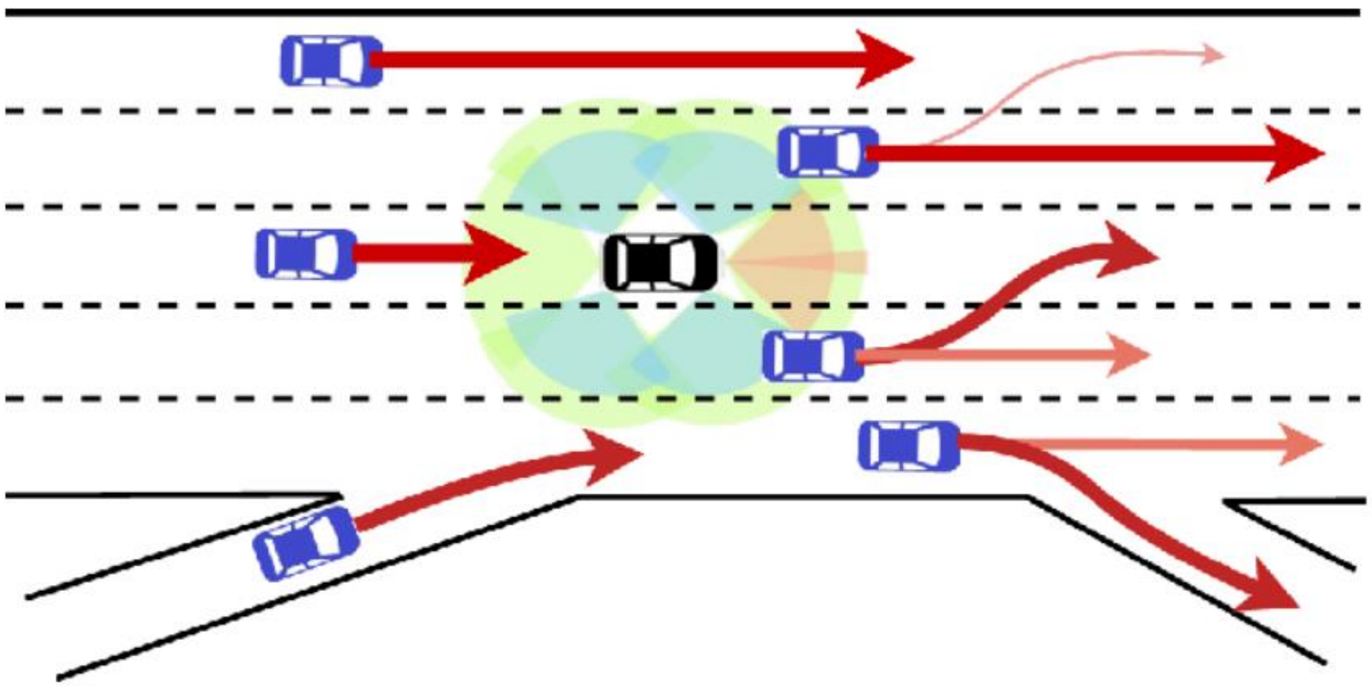
Trajectory Prediction: Overview

- Task Overview



Trajectory Prediction: Definition

- Task Definition
  - Given HD Map, neary agents' history states.
  - Output one or more agents' future states.



Trajectory Prediction

- Non-learning methods: very old
  - Constant Velocity
- Learning-based methods:
  - Raster: too much computation as image
    - Square growth to retrevial radius
  - Vector-based: each entity as a vector
    - More popular and efficient
      - Transformer-based
      - Gnn-based

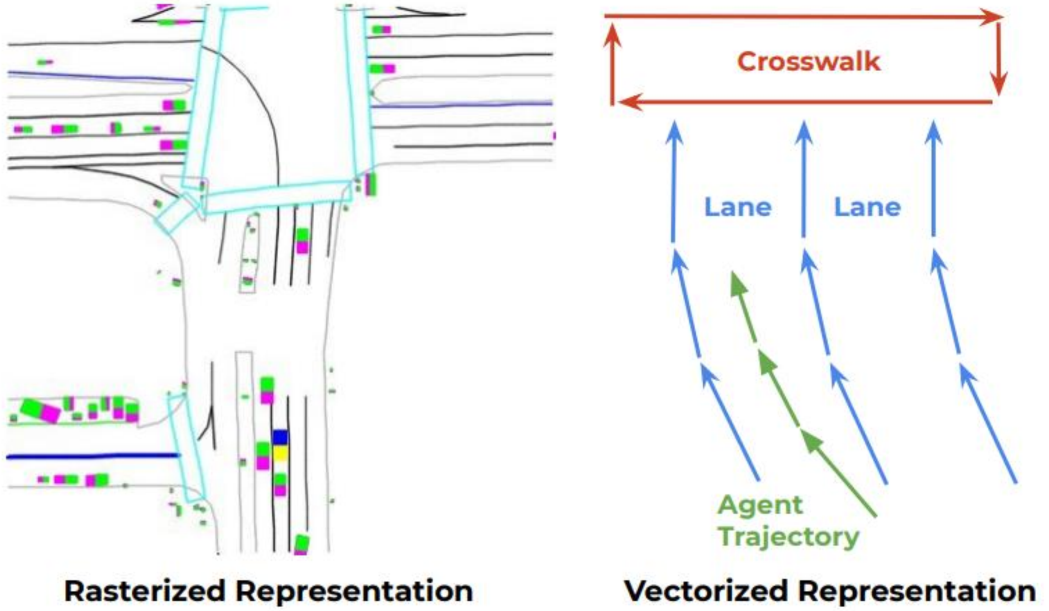


Figure 1. Illustration of the rasterized rendering (left) and vectorized approach (right) to represent high-definition map and agent trajectories.

VectorNet, *CVPR'20*

SmartRefine: A Scenario-Adaptive Refinement Framework for Efficient Motion Prediction

Yang Zhou[1*] Hao Shao[1,2*] Letian Wang[3]
Steven L. Waslander[3] Hongsheng Li[2,4,5] Yu Liu[1,5] ✉

[1]SenseTime Research [2]CUHK MMLab [3]University of Toronto
[4]CPII under InnoHK [5]Shanghai Artificial Intelligence Laboratory

[Published in *CVPR 2024*]

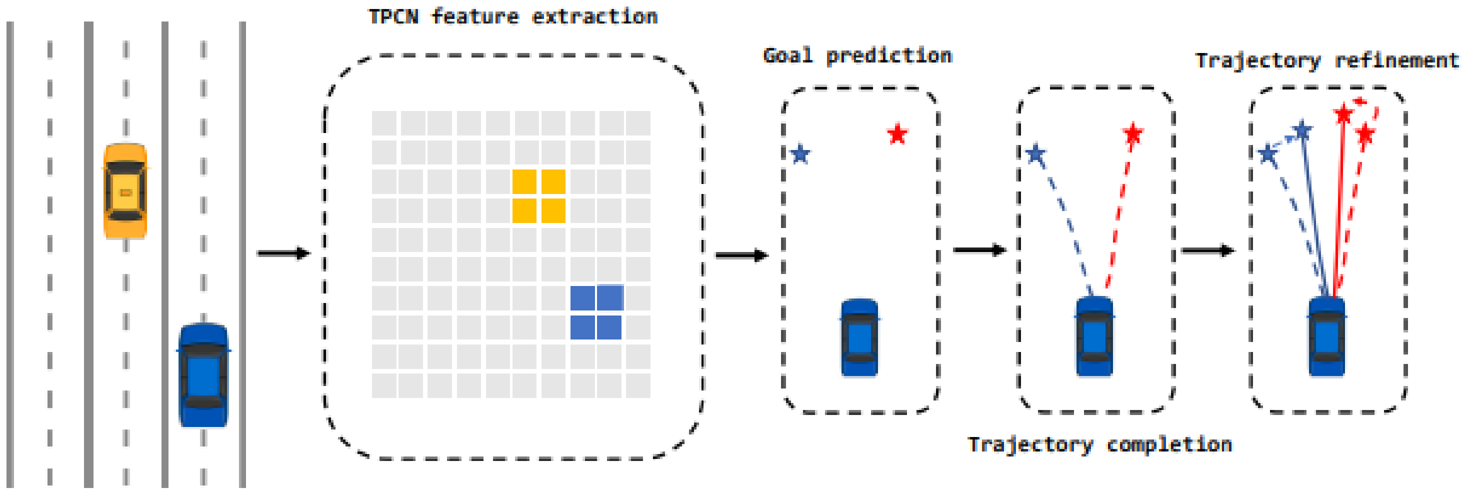
Background: What’s refinement and why?

- Refinement: Two-stage prediction methods
  - Stage 1: Given HD Map, past states, output future states.
  - Stage 2: Using predicted future states, output delta offset thus make a new trajectory
    - Hd map + x -> y
    - Y + your refinement -> delta_y   y_new = y + delta_y

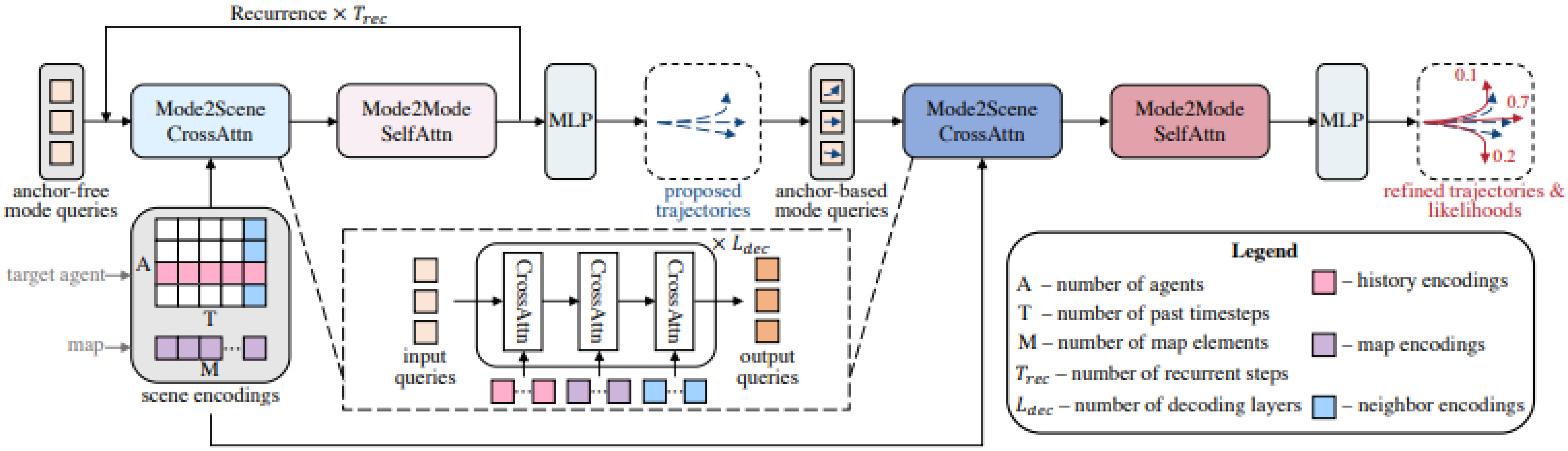
- Why refinement:
  - Corase to fine: easier to learn, distribution mapping / shift
  - Corase trajectory can be useful for aggregate specific context information

Background: Existing Refinement Methods

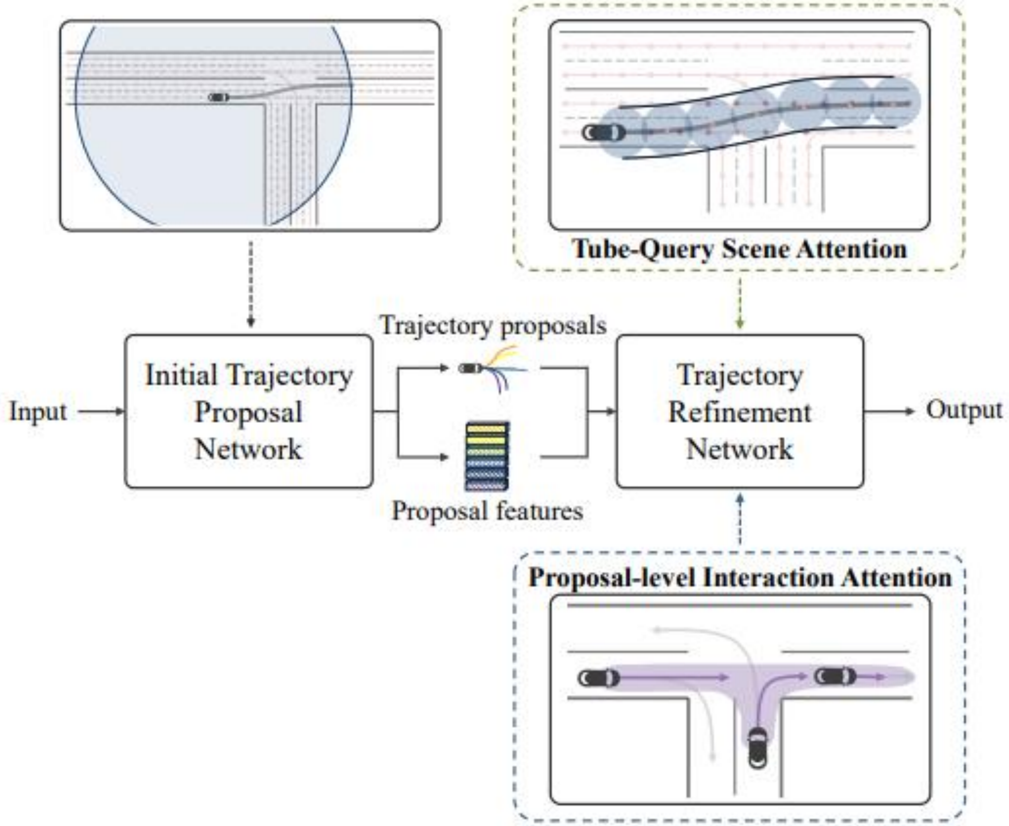
DCMS or MISC, *ICCV'23*



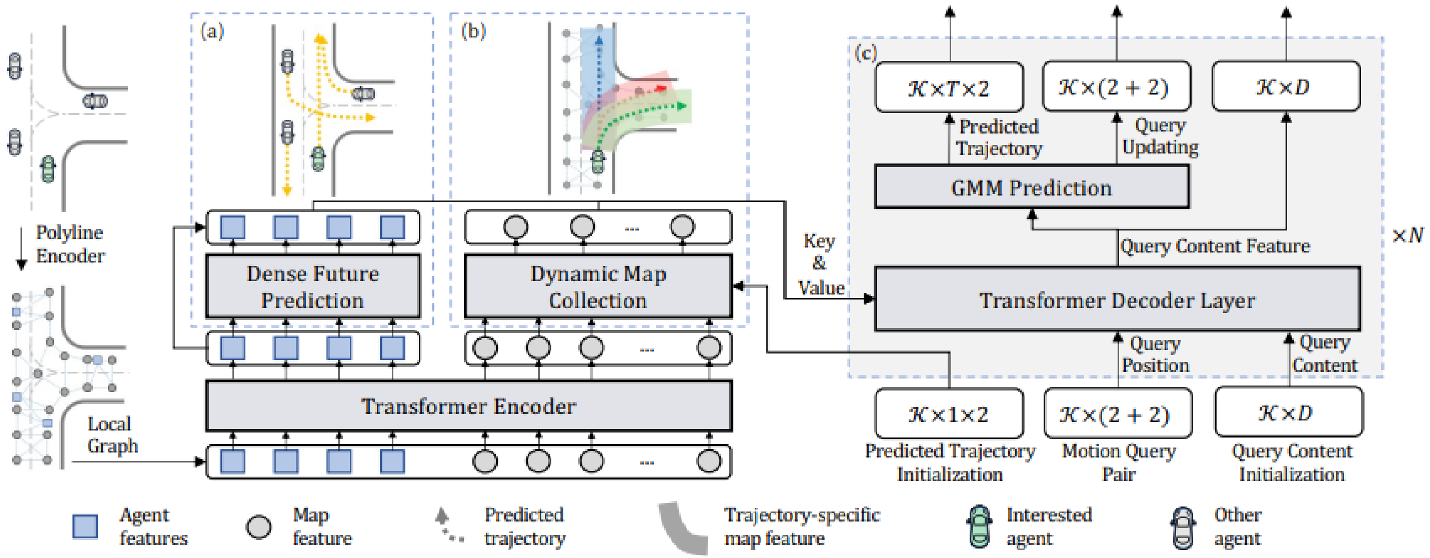
QCNet, *CVPR'23*



Background: Existing Refinement Methods



R-Pred, *ICCV'23*



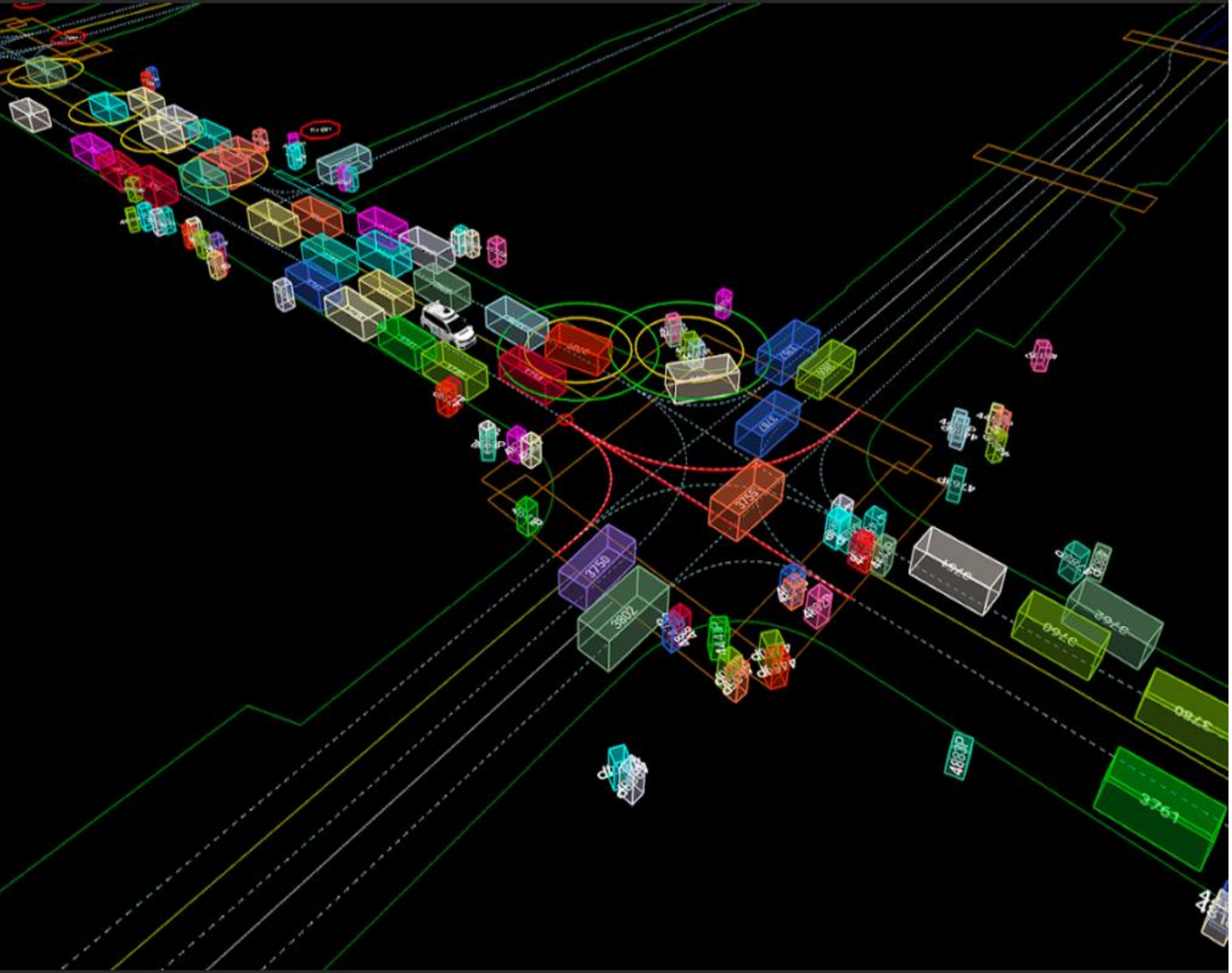
MTR, *NIPS'22*

Background: Existing Refinement Methods

- Summary
- ❌Coupled with Backbone
- ❌Fixed refinement setting

	Refine	Context Selection/Encoding			Refinement Iteration		
		All	Fixed Strategy	Adaptive Strategy	Single	Multiple	Scenario Adaptive
TNT [38]	✗						
GoalNet [37]	✗						
GANet [32]	✗						
ProphNet [33]	✗						
DCMS [36]	✔	✗	✗	✗	✔	✗	✗
QCNet [40]	✔	✔	✗	✗	✔	✗	✗
R-Pred [5]	✔	✗	✔	✗	✔	✗	✗
MTR [25]	✔	✗	✔	✗	✗	✔	✗
SmartRefine (ours)	✔	✗	✗	✔	✗	✗	✔

Table 1. A comparison between the proposed SmartRefine framework and previous methods in terms of 1) whether refinement is conducted; 2) how the context selection and encoding is conducted; 3) how to determine the number of refinement iterations.



Human Driver

- Suppose you are driving:
    - process 50m/100m all agents' past behavior? No
    - Keep interction of all agents's at every tiimestep?
    - No
- Only critical/task relevant context information

Learn from Human Driver

- Human drivers can **easily predict surrounding agents' future behaviors**, even if they confront a daunting amount of context information.
- As implied by neuroscience, humans' efficient reasoning capabilities benefit from their **selective attention mechanism**, which identifies compact context information **critical to the task** for efficient reasoning.
- Similarly, motion prediction models are shown to be able to produce high-quality predictions with only a few critical context elements provided, such as **only giving the ground-truth future reference lanes**.
- Therefore, if we can identify the critical context elements, and **aggregate more information** from these critical inputs to **further refine the predictions**, both the computational efficiency and prediction performance can be significantly improved.

Motivation: How to improve refinement?

- Aim: Decoupled from backbone
  - Only need general interface: trajectories and trajecrory embeddings

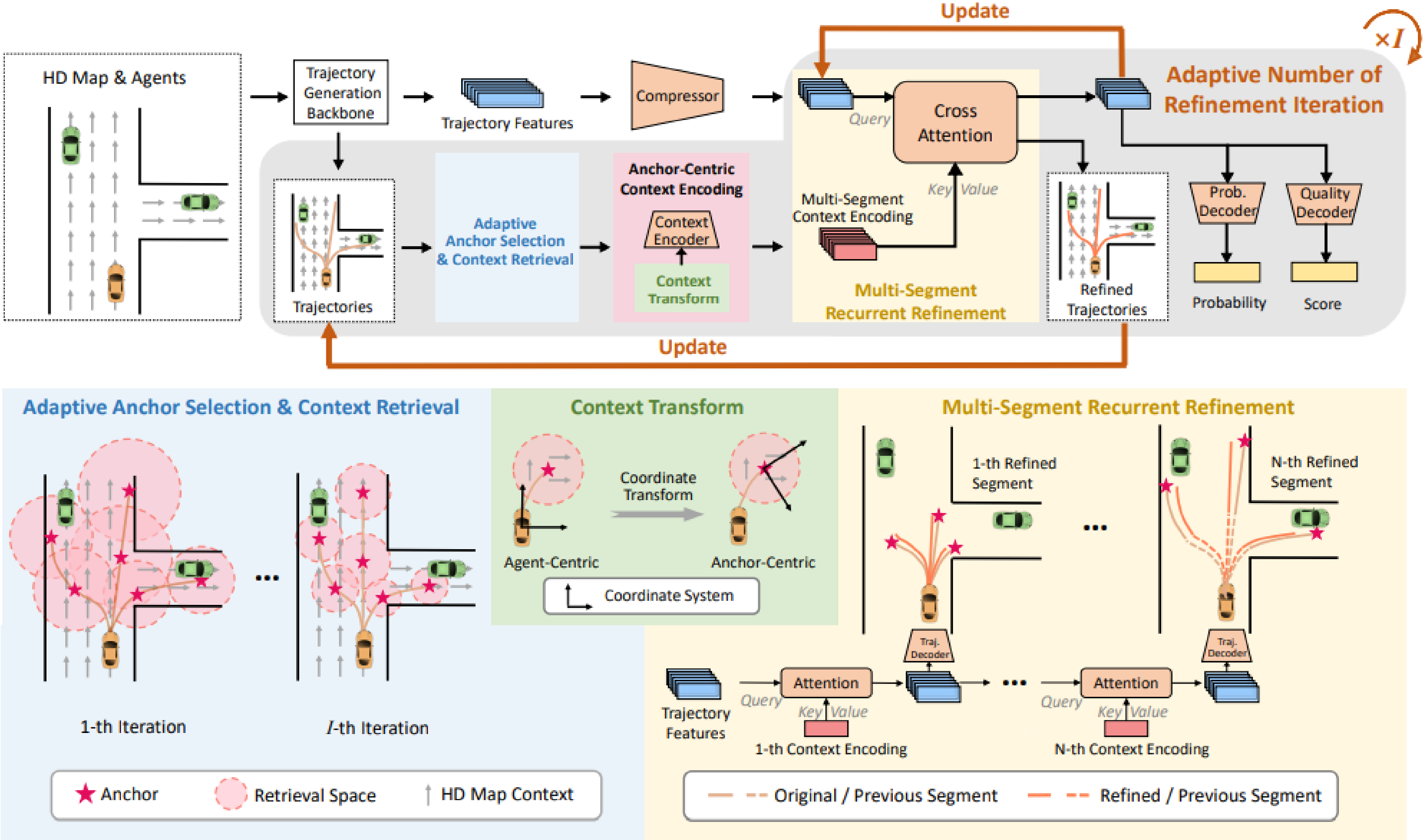
- Aim: selectively choose critical context
  - Like smart human driver
  - Computation constrain

- Aim: From one iteration/multiple itrtration to adaptive iteration
  - One iteration: potentially not enough
  - Multiple iteration: latency concern

Contribution

- We introduce SmartRefine, a **scenario-adaptive** refinement method to effectively **enhance prediction accuracy with limited additional computation**.
- We propose a **generic and flexible** refinement framework, which can be easily integrated into most prediction methods.
- We conduct **extensive experiments** on Argoverse and Argoverse 2 datasets and show that SmartRefine improves the accuracy with little additional computation.
- Comprehensive studies are also conducted to **ablate design choices and explore the mechanism** behind multi-iteration refinement.

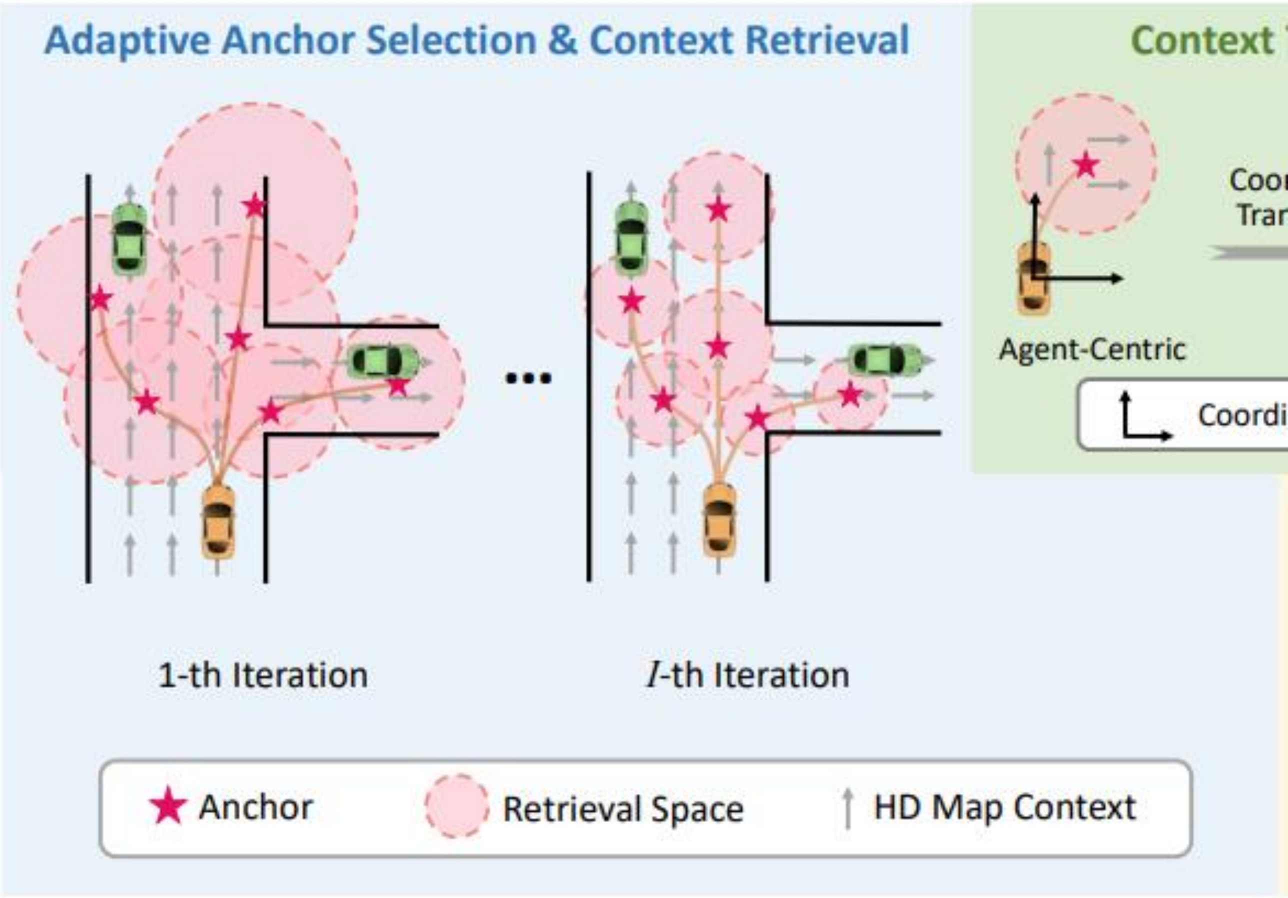
Methods: Pipeline



Methods: Adaptive Anchor/Context Selection

- Anchor Selection
  - Naïve option: last point
  - Other extreme: all points
  - Ours: N points with N segments

- Contexst Retrieval
  - Previous: fixed radius
  - Ours: adaptive to two conditions
    1. Refine iteration
    2. Agent's speed around the ancor

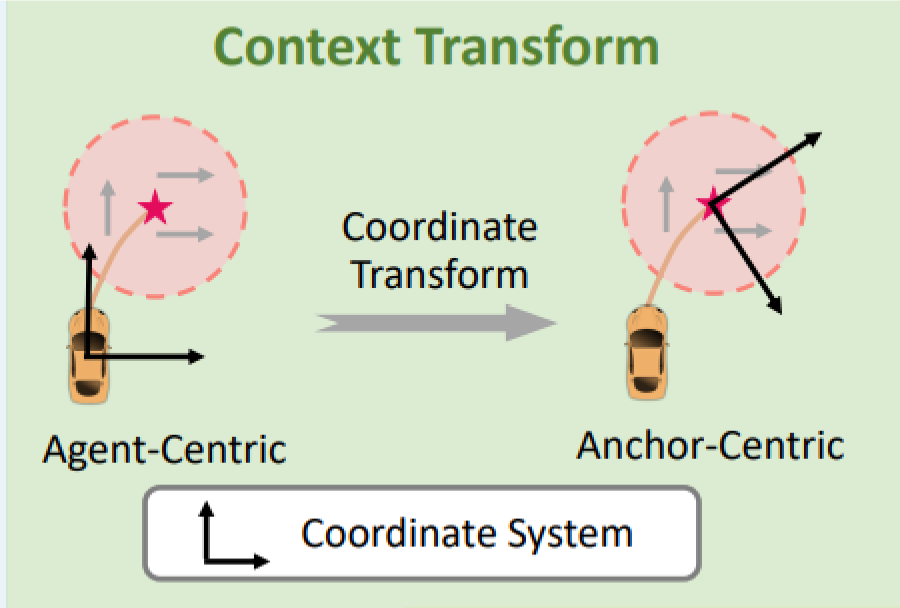


Methods: Anchor-Centric Context Encoding

- Target-centric to anchor-centric

1. Better caapture future trajectory details (yaw, position)

2. Anchors are dynamically changing (thus vary context)



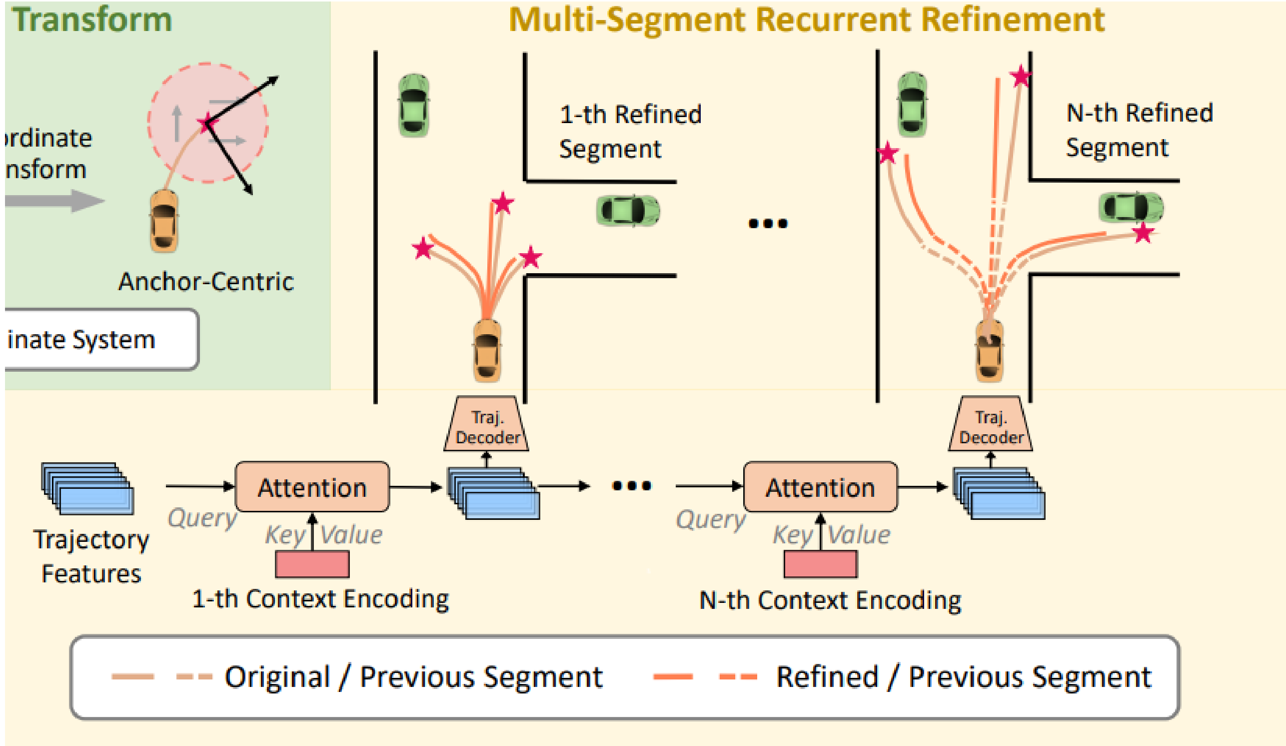
Methods: Recurrent and Multi-Iteration Refinement

- Recurrent Refinement **in each refine iteration**

- Query: trajectory feature
- K/V: anchor around context

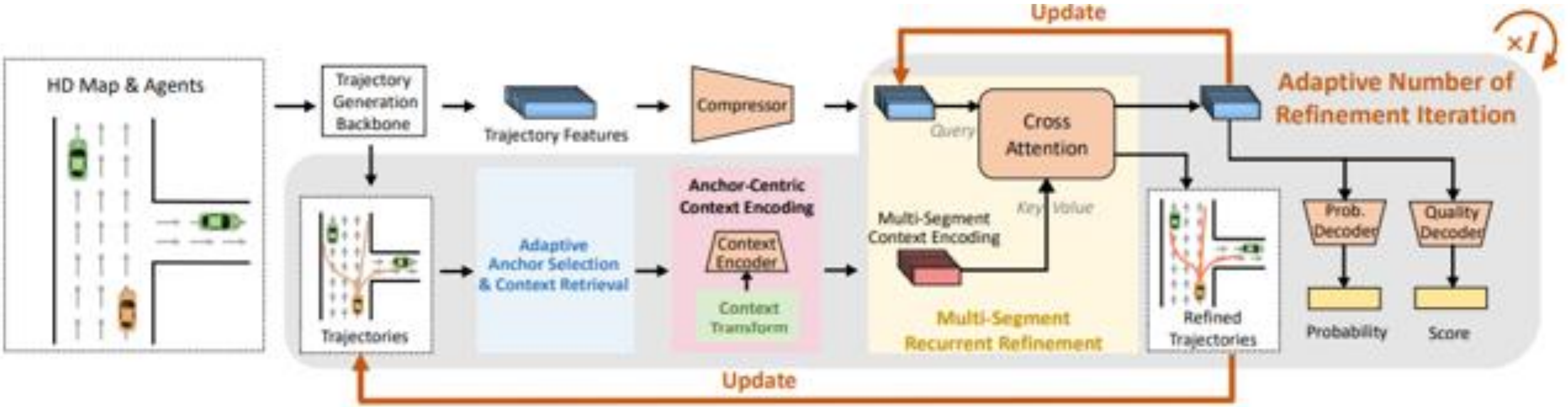
- Ouput: trajectory delta offset

- Recurrent N segmets



Methods: Recurrent and Multi-Iteration Refinement

- Multi-Iteration Refinement



Methods: Adaptive Number of Refinement Iterations

- Aim: A measure to understand the potential of refinement
  - Also need to be trainable
- Quality Score Design:
  - Decide whether another refinement is needed

Quality Score Design. At the training stage, since we have access to the ground-truth trajectory and predicted trajectory of all refinement iterations, an intuitive measure for the quality of the predicted trajectory in iteration i is $d_{max} - d_i$, where d_{max} denotes the largest predicted error among all iterations. A small d_i means that the current prediction has already been improved from the largest prediction error d_{max}, and thus of high quality. However, this design as the quality score can be unstable as it lies in a big range and can vary a lot in different scenarios. We then normalize $d_{max} - d_i$ with $d_{max} - d_{min}$, where d_{min} denotes the smallest predicted error among all iterations. Thus the final quality score lies in between [0,1] and is designed as

$$q_i = \frac{d_{max} - d_i}{d_{max} - d_{min}} \tag{1}$$

Methods: Adaptive Refinement Iteration

- Aim: How to use quality score to achieve adaptive refinement.

- 3 rules:
  - Good enough
  - Trajectory becomes worse
  - Reach max iteration

Algorithm 1 Adaptive Inference with SmartRefine

Input: Backbone model f_b, refinement model f_r, quality score decoder f_d, agents history trajectories $\mathbf{s}_h$, scene context $\mathbf{c}$, and score threshold $\bar{q}$, maximum refinement iteration at inference I'.

Output: Target agent's future trajectories $\mathbf{s}_f$ and corresponding probabilities $\mathbf{p}$.

```
s_f^0, h_f^0, p^0 = f_b(s_f^0, c)                 % Backbone prediction
q^0 = f_d(h_f^0)                                  % Initial quality score
if q^0 > q̄ then                                   % No need for refinement
    Return s_f^0, p^0
for i = 1, 2, ..., I' do                          % Multi-iteration refine
    Adaptively select anchors and contexts
    Adpatively encode contexts c^{i-1} (anchor-centric)
    Multi-segment recurrent refinement:
        δs_f^i, h_f^i, p^i, q^i = f_r(s_f^{i-1}, h_f^{i-1}, c^{i-1})
        s_f^i = s_f^{i-1} + δs_f^i
    if q^i < q^{i-1} then                         % Terminate refinement
        Return s_f^i, p^i
Return s_f^i, p^i                                 % Reach max refinement iteration
```

Training

- Standard training procedure plus score loss.

- We use L1 loss to regress quality score.

3.3. Training Loss

The training of our model considers three loss terms, and uses hyper-parameter α to balance them:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}} + \alpha \cdot \mathcal{L}_{\text{score}} \tag{2}$$

where $\mathcal{L}_{\text{cls}}$ denotes the cross-entropy classification loss for predicting the probability of the multi-modal trajectories. For the regression loss $\mathcal{L}_{\text{reg}}$, as our model predicts a Laplace distribution for each time step's waypoint, we calculate the negative log-likelihood of the ground-truth trajectory in the predicted distribution. Note that among all predicted multi-modal trajectories, only the modal closest to the ground truth is considered for the regression loss. For the quality score loss $\mathcal{L}_{\text{score}}$, in the training stage, we fix the refinement iteration as I and each iteration will output a quality score. Thus for each iteration i, we calculate the ℓ_1 loss between the predicted quality score $\hat{q}_i$ and labeled quality score q_i (see Sec 3.2.4), and average the loss over all iterations:

$$\mathcal{L}_{\text{score}} = \frac{1}{I+1} \sum_{i=0}^{I} \|\hat{q}_i - q_i\|_1 \tag{3}$$

Similarly, the score loss also only considers the modal closest to the ground truth.

Experiments: Settings

- Datasets: argo1 & argo 2
  - 205k / 39k / 78k 200k / 25k / 25k

- Metrics: minFDE minADE MR over 6 modalities

- Methods: SmartRefine plug into existing baselines
  - HiVT (CVPR'22) ProphNet (CVPR'23) mmTransformer (CVPR'21)
  - DenseTNT (ICCV'21) QCNet no ref QCNet (CVPR'23)

Experiments: Quantutative Result

- Val Set
- SmartRefine can consistently improve the prediction accuracy of all considered methods with limited added parameters, Flops, and latency.

Dataset	Method	minFDE ↓	minADE ↓	MR ↓	#Param.(M) ↓	Flops(G) ↓	Latency(ms) ↓
Argoverse	HiVT [39]	0.969	0.661	0.092	2.5	2.6	54±4.0
	HiVT w/ Ours	0.911	0.646	0.083	2.7	2.7	67±8.4
	Prophnet* [33]	1.004	0.687	0.093	15.2	7.8	59±1.7
	Prophnet w/ Ours	0.967	0.675	0.092	15.4	7.9	71±6.2
	mmTransformer [14]	1.081	0.709	0.102	2.6	1.2	15±4.8
	mmTransformer w/ Ours	1.023	0.692	0.094	2.8	1.3	27±9.7
Argoverse 2	DenseTNT [10]	1.624	0.964	0.233	1.6	3.6	1,075±199
	DenseTNT w/ Ours	1.553	0.834	0.221	1.9	4.0	1,099±212
	QCNet (no ref)	1.304	0.729	0.164	5.5	47.0	338±53
	QCNet (no ref) w/ Ours	1.258	0.718	0.157	5.8	47.4	363±67
	QCNet [40]	1.253	0.720	0.157	7.7	55.8	392±54
	QCNet w/ Ours	1.240	0.716	0.156	8.0	56.2	418±68

Experiments: Quantutative Result

- Test Set

Dataset	Method	minFDE ↓	minADE ↓	MR ↓
Argoverse	HiVT [39]	1.17	0.77	0.13
	HiVT w/ Ours	1.13	0.77	0.12
	ProphNet* [33]	1.30	0.85	0.14
	Prophnet* w/ Ours	1.21	0.81	0.13
	mmTransformer [14]	1.34	0.84	0.15
	mmTransformer w/ Ours	1.24	0.81	0.14
Argoverse 2	DenseTNT [10]	1.66	0.99	0.23
	DenseTNT w/ Ours	1.59	0.85	0.22
	QCNet (no ref)	1.29	0.65	0.16
	QCNet (no ref) w/ Ours	1.24	0.64	0.15
	QCNet [40]	1.24	0.64	0.15
	QCNet w/ Ours	1.23	0.63	0.15

Rank	Method	minFDE ↓	minADE ↓	MR ↓
1	SEPT-iDLab (SEPT)*	1.15	0.61	0.14
2	GACRND-XLAB (XPredFormer)*	1.20	0.62	0.15
3	QCNet-AV2 (QCNet)	1.19	0.62	0.14
4	MTC (MTC)*	1.17	0.61	0.14
5	Mingkun Wang	1.19	0.62	0.14
6	AnonNet (AnonNet)*	1.23	0.63	0.15
7	ls (TraceBack)*	1.20	0.64	0.14
8	SmartRefine (ours)	1.23	0.63	0.15
-	QCNet (no ensemble)	1.24	0.64	0.15
-	GANet (published version)	1.35	0.73	0.17

- Argo 2 leaderboard at the time of paper submission
- *: not published.
- Other two: improved version

Experiments: Adaptive Refinement Iterations

- Setting
  - Change score threshold
  - Change max iteration number

- Adaptive strategy maintain performance with less refinement iterations.

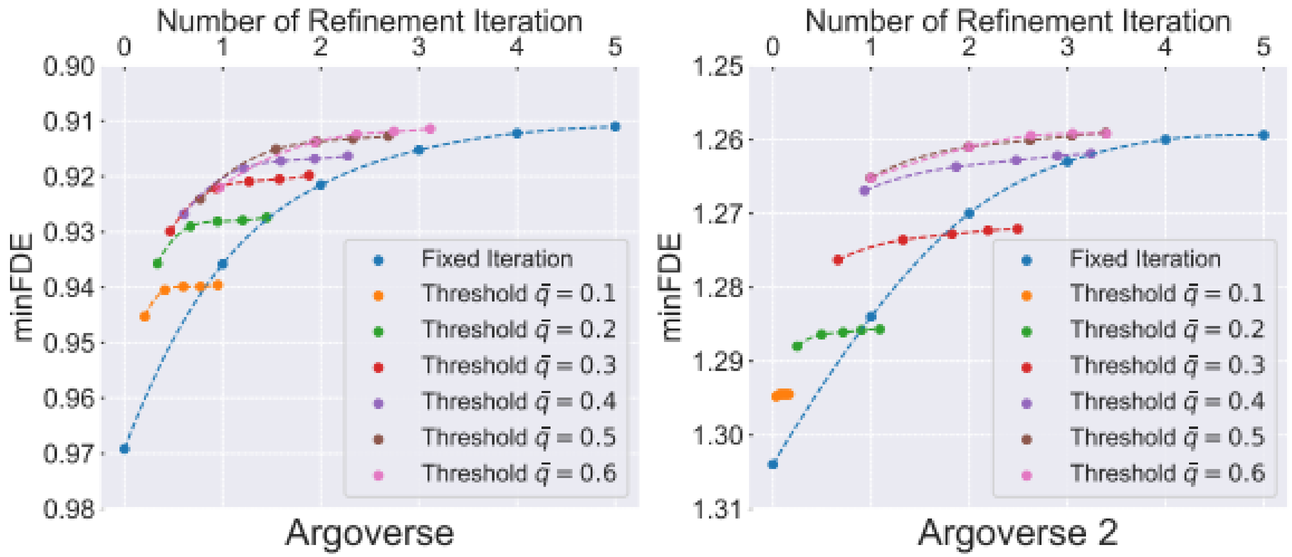


Figure 2. Comparison between the fixed and adaptive number of refinement iterations. For the adaptive methods, We tested different quality score thresholds $\bar{q}$ mentioned in Algorithm 1.

Experiments: Comparison with other methods

- Our method which utilizes adaptive refinement achieves **the best trade-off** in minFDE and latency.

Backbones Datasets	Metrics	Different Ideologies of Refinement Techniques					
		no ref	DCMS[1]	QCNet[1]	R-Pred[1]	MTR^M	Ours^A
HiVT Argo	minFDE	0.969	0.958	0.933	0.929	0.915	0.911
	Latency	54±4.0	55±4.4	64±5.1	62±5.9	92±9.4	67±8.4
Prophnet Argo	minFDE	1.004	0.996	0.984	0.981	0.968	0.967
	Latency	59±1.7	60±2.4	68±3.2	65±3.1	88±5.9	71±6.2
mmTransformer Argo	minFDE	1.081	1.066	1.048	1.045	1.022	1.023
	Latency	15±4.8	16±5.5	22±5.6	21±5.5	51±8.4	27±9.7
DenseTNT Argo 2	minFDE	1.624	1.601	1.563	1.576	1.553	1.553
	Latency	1,075±199	1,076±199	1,133±217	1,085±209	1,125±213	1,099±212
QCNet (no ref) Argo 2	minFDE	1.304	1.293	1.253	1.274	1.256	1.258
	Latency	338±53	339±53	392±54	348±55	387±62	363±67

Table 8. Comparison of refinement methods. 1/M/A denotes one-iteration, multi-iteration, and adaptive-iteration refinement methods respectively. Our method which utilizes adaptive refinement achieves the best trade-off in minFDE and latency.

Experiments: Ablation Studies

- Anchors: 2 is good enough
- Context transformation helps prediction accuracy
- Our adaptive retrieval strategy save computation.

#Anchor numbers	minFDE	#Param.
1	0.928	134K
2	0.911	207K
3	0.911	280K
5	0.915	433K
6	0.916	509K

Table 4. Ablation study on the number of anchors.

Context Encoding	minFDE
Agent-Centric	0.941
Anchor-Centric	0.911

Table 5. Ablation study on how the contexts are encoded.

	Retrieval Radius	minFDE	Flops (M)
Fixed Radius	50	0.926	2,297
	20	0.923	722
	10	0.921	325
	2	0.930	58
Adaptive Radius	R_{max}=10, R_{min}=2, linear	0.911	245
	R_{max}=10, R_{min}=2, exp	0.911	130

Table 6. Ablation study on the context retrieval radius. Linear and exp denote different ways to decay the radius.

Discussion: Why adaptive refinement?

- **Not every trajectory benefits from multi-iteration refinement.**

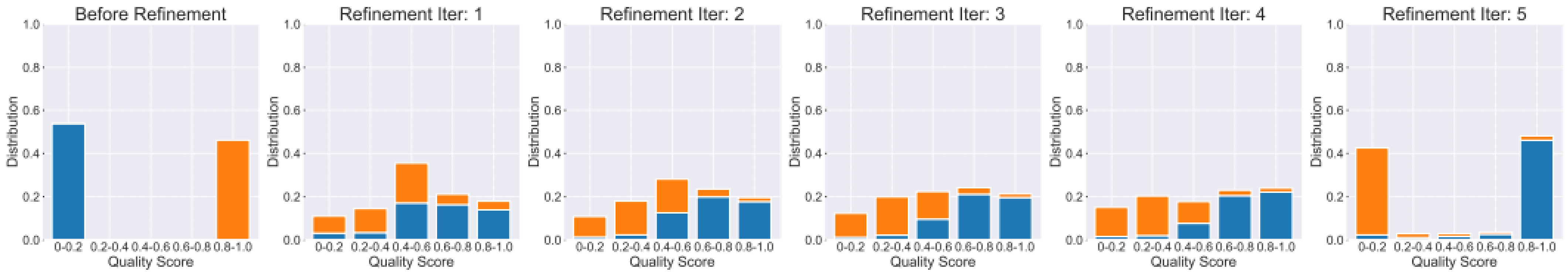
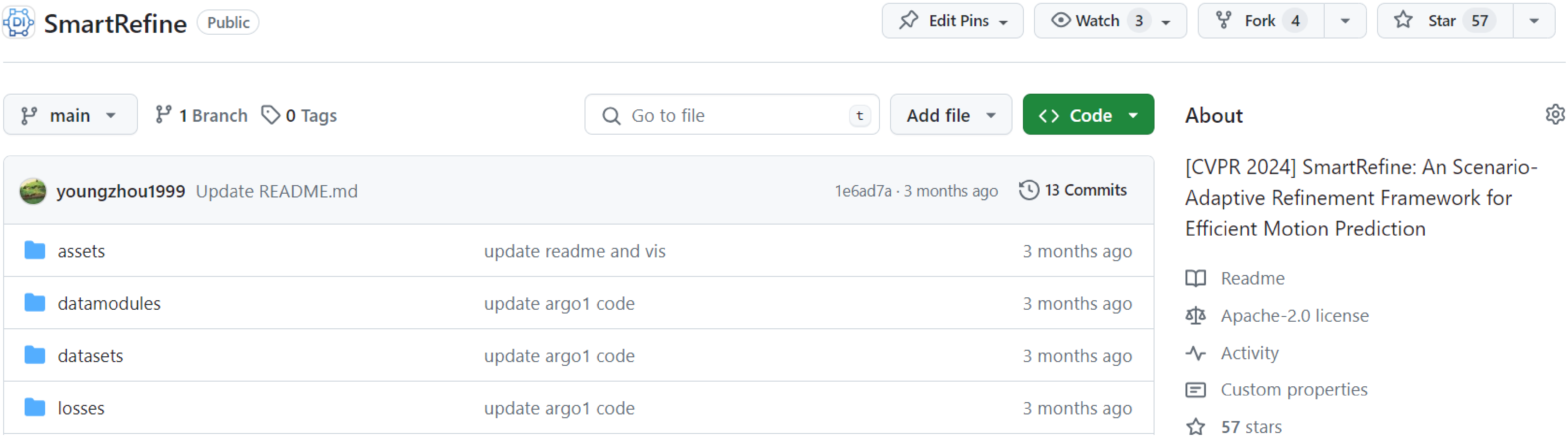


Figure 3. A study to understand the mechanism behind the refinement. Specifically, We mark the quality score distribution of the predictive trajectory before refinement, and track how the quality score changes along the multi-iteration refinement. We can see while the overall performance is improved, not every trajectory benefits from refinement, which implies the necessity of adaptive refinement. See Sec. 4.4 for detailed discussions.

Code available

- https://github.com/opendilab/SmartRefine



Open Thoughts: Refinement

- Suitable for Practical situation
  - Refinement utilize much less computation
  - Specific context retrieval
  - Online improve trajectories
  - Case by case: adaptive to scenarios

1. Offline prediction, online refinement if needed.
2. case by case improvement： traffic scenarios， long-tail scenarios and so on
3. Multi-agents’ refinemmet: refine based on other agents’ behavior jointly.

Takeaway

- Two stage trajectory prediction.
- Adaptive refinement with quality score.
- Efficient configuration for potentially practical use.