# Mind the map! Accounting for existing maps when estimating online HDMaps from sensors.

Rémy Sun, Li Yang, Diane Lingrand, Frédéric Precioso

# Mind the map! Accounting for existing maps when estimating online HDMaps from sensors.

Rémy Sun Li Yang Diane Lingrand Frédéric Precioso
Université Côte d'Azur, Inria, CNRS, I3S, Maasai, Nice, France
{firstname.lastname}@univ-cotedazur.fr

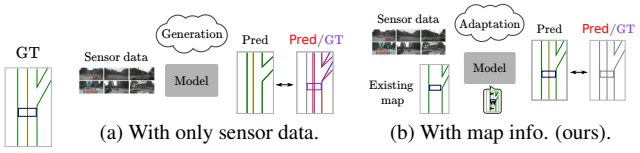
## Abstract

*While HDMaps are a crucial component of autonomous driving, they are expensive to acquire and maintain. Estimating these maps from sensors therefore promises to significantly lighten costs. These estimations however overlook existing HDMaps, with current methods at most geolocalizing low quality maps or considering a general database of known maps. In this paper, we propose to account for existing maps of the precise situation studied when estimating HDMaps. To prove this, we identify 3 reasonable types of useful existing maps (minimalist, noisy, and outdated). We then introduce MapEX, a novel online HDMap estimation framework that accounts for existing maps. MapEX achieves this by encoding map elements into query tokens and by refining the matching algorithm used to train classic query based map estimation models. We demonstrate that MapEX brings significant improvements on the nuScenes dataset. For instance, MapEX - given noisy maps - improves by 38% over the MapTRv2 detector it is based on and by 8% over the current SOTA.*

## 1. Introduction

Autonomous Driving [13,26] represents a complex problem that promises to significantly change how we interact with transportation. While full vehicle automation still seems quite a ways away [44], partially autonomous vehicles now populate a number of road systems in the world [26]. These vehicles need to process a wealth of information to function, from the raw sensor data [16] to elaborate maps of road networks [13,31].

High Definition maps (HDMaps), in particular, represent a crucial component of the research on self-driving cars [12,13] (see Fig. 1 for a few simple examples of maps, with road boundaries represented by green polylines, lane dividers by lime polylines and pedestrian crossings by blue polygons). Although maps are not a typical input of neural networks, they contain necessary information to help the car understand the world it must navigate. As such, significant efforts have gone into incorporating this new type of data into solutions [13, 40]. These efforts have shown HDMaps' remarkable benefits both for fundamental problems like Object Detection [19] to precise trajectory forecasting problems [31,40].




(a) With only sensor data. (b) With map info. (ours).

Figure 1. We propose to use existing map information - even if inexact - to estimate better online HDMaps from sensor inputs. In doing so, we simplify the problem from generating maps using only sensors to re-using available existing maps aided by sensors.

These maps are however expensive to acquire and maintain, requiring precise data acquisition and exacting human labeling [12, 23]. There has therefore been a strong push [6, 32] to approximate HDMaps from sensor data. Seminal work [35] has proved even rough estimated maps useful for trajectory forecasting.

While recent methods like MapTRv2 [33] have become proficient at estimating HDMaps from raw sensors, we feel they overlook very useful and nearly always available data: existing maps. We posit here that outdated or lower quality maps should usually be available and prove to significantly improve the estimated HDMaps as illustrated in Fig. 1. Indeed, even "map-free" models tend to use lower-quality satnav maps [1], and estimated maps could always be available as long as a vehicle went through a place once.

In this paper, we explore the central postulate that **even inaccurate existing maps improve the estimation of HDMaps** from raw sensors. After providing some context on our method and the field in Sec. 2, we propose two distinct technical contributions to study this idea: In Sec. 3, we outline reasonable scenarios under which an inaccurate map can be available along with practical implementations, and in Sec. 4, we propose MapEX, an architecture that can

generate HDMaps from sensor data while accounting for existing map information. Finally, we present results in Sec. 5 with experiments on the nuScenes dataset [5].

*Contributions* We detail three contributions in this paper:

- We propose to **account for existing map information** when estimating online HDMaps from sensor data.
- We provide practical settings to evaluate this idea by proposing **reasonable scenarios** under which existing maps are not perfect. We also provide realistic implementations of these scenarios and the code for the nuScenes dataset.
- We introduce **MapEX**, a new query based HDMap estimation framework that can **incorporate existing map information when estimating an online HDMap from sensors**. In particular, we introduce with MapEX both a novel way to incorporate existing map information with non-learnable existing (EX) queries, and a way to ensure the model uses this information by pre-attributing predictions to known ground truth correspondences during training.

*Impact* We believe they are of interest to the field as:

- Using existing maps drastically **lowers the bar** needed to obtain good and cost-effective map estimations. In one scenario where we use HDMaps with noisy (or "shifted") map element positions, for instance, MapEX reaches a 84.8% mAP score which is an improvement of 38% over the MapTRv2 detector it is based on. This is also a 8% improvement over the state-of-the-art set by MapNeXt [30] using a foundation model image backbone [47] (vs. our ResNet-50 backbone). As estimated maps become more complex, existing maps will become more and more crucial to good and cost-effective performance.
- MapEX is the **first method to directly integrate an existing map** corresponding precisely to the sensor data from the given location (i.e. the input sample) to guide online HDMap estimation and subsequent work [3] both builds on our work and validates our findings. To the best of our knowledge, this is a blind spot of the literature, with previous works only considering geolocalized satellite maps [14], SDMaps [2, 37], or trying to retrieve an existing map similar to the sample from a pre-computed database [48]. None of these methods leverage sensor data to correct a flawed existing map.
- We provide the implementation of existing map scenarios for online HDMap estimation following three realistic scenarios. Our implementations very importantly provide both (flawed) existing HDMaps and the true HDMap for each sample. This is not the case in the existing *Trust but Verify* [27] map change detection dataset which only provides the existing HDMap and a label as to whether a change has occurred.

*Map nomenclature* We work on **local** $30m \times 60m$ maps restricted to a sample's surroundings. **True** maps are the ground truth maps, **existing** maps are the maps available as inputs, **predicted** maps are the maps estimated by a model.

## 2. Related Work

We provide here some brief context on HDMaps in autonomous driving. We begin by discussing HDMap's use in trajectory forecasting, before discussing their acquisition. We then discuss online HDMap estimation itself.

*HDMaps for trajectory forecasting* Autonomous Driving requires a lot of information about the world vehicles are to navigate. This information is typically embedded in rich HDMaps given as input to modified neural networks [17, 40]. HDMaps have proven critical to the performance of a number of modern methods in trajectory forecasting [10, 40] and other applications [19]. In trajectory forecasting in particular, it is remarkable that some methods [34, 36] explicitly reason on a representation of the HDMap and therefore absolutely require access to a HDMap [43]. [35] reports a 10% drop in performance for a common forecasting technique [36] when applied without an informative HDMap. [50] reports even more dramatic drops in performance for other well known methods.

*HDMap acquisition and maintenance* Unfortunately, HDMaps are expensive to acquire and maintain [12, 23]. While HDMaps used in forecasting are only a simplified version containing map elements (lane dividers, road boundaries, ...) [28, 33] and leave out much of the complex information in full HDMaps [12], they still require exceedingly precise measurements (on the scale of tens of centimeters) [12]. A number of companies have therefore been moving towards a less exacting standard with Medium Definition Maps (MDMaps) [18], or even simpler Standard Definition Maps (SDMaps) such as satellite navigation maps, Google Maps, etc [1]. Crucially, MDMaps - with their precision of a few meters - would be a good example of an existing map giving valuable information for the online HDMap generation process. Our map **Scenario 2a** explores an approximation of MDMaps.

*Online HDMap estimation from sensors* Online HDMap estimation [6] has therefore emerged as a promising alternative to manually curated HDMaps. While some works [6, 7, 51] focus on predicting virtual map elements, i.e. lane centerlines, the standard formulation introduced by [28] focuses on more visually recognizable map elements: lane dividers, road boundaries and pedestrian crossings. Probably because visual elements are easier to detect by sensors,

this latter formulation has seen rapid progress over the last years [11, 32, 35]. Interestingly, the latest such method - MapTRv2 [33] - does offer an auxiliary setting for detecting virtual lane centerlines. This suggests a natural convergence towards the more complex settings comprising a multitude of additional map elements (traffic lights, ...) [29, 45]. Nevertheless, **the standard formulation from [28] remains the gold standard** when evaluating the usefulness of additional information such as learned global feature maps [49], satellite views [14], or SDMaps [2]. We thus keep to this standard problem formulation to demonstrate the use of existing map information.

Our work is adjacent to the commonly studied map change detection problems [4, 38] that aim to detect a change in a map (e.g. crossings). While rooted in more classical statistical techniques [38], a few efforts have been made to adapt them to deep learning [4, 21]. Notably, the Argoverse 2 *Trust but Verify* (TbV) dataset [27] was recently proposed for this problem (see Appendix Sec. 8). This however differs substantially from our approach as we do not try to correct small mistakes on an existing map after aggregating from a fleet of vehicles [24, 39]. Instead we aim to generate accurate online HDMaps with the help of an existing - possibly very different - map, which is made possible by the modern online HDMap estimation problem. Therefore, **we do not only correct small mistakes in maps but propose a more expressive framework that accommodates any change** (e.g. distorted lines, very noisy elements).

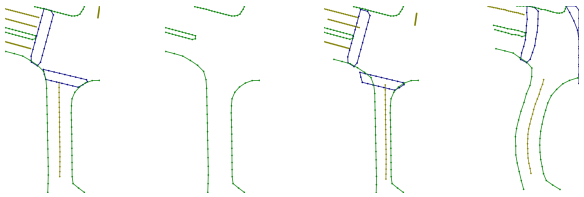
# 3. What Kind of Existing Map Could We Use?

We make the central claim that accounting for existing maps would benefit online HDMaps estimation. To prove it, we point out some of the many reasonable scenarios under which imperfect existing maps can appear. After defining our HDMap representations in Sec. 3.1 and our general approach in Sec. 3.2, we consider three main possibilities: only road boundaries are available (Sec. 3.3), the maps are noisy (Sec. 3.4), or they have changed substantially (Sec. 3.5).

## 3.1. HDMap Representation

We adopt the standard format used for online HDMap estimation from sensors [28, 32]. We consider HDMaps to be made of three types of polylines (as represented on Fig. 2a): road boundaries, lane dividers and pedestrian crosswalks with same colors as previously green, lime, and blue respectively. We follow [32] by representing these polylines as sets of 20 evenly spaced points for our map generator, with upsampled versions for evaluation.

While complete HDMaps are much more complex [12] and more intricate representations have been proposed [7], the aim of this work is to study how to account for existing map information. As such we restrict ourselves to the most commonly studied formulation (road boundaries, lane dividers and pedestrian crosswalks), but our approach will be directly applicable to the prediction of more map elements [7], finer polylines [11, 42] or rasterized objectives [52].



(a) Ground Truth: True HDMap. (b) **Scenario 1**: only boundaries. (c) **Scenario 2a**: element shifts. (d) **Scenario 3a**: outdated maps.

Figure 2. Examples of HDMaps generated by MapModEX.

## 3.2. MapModEX: Simulating Imperfect Maps

As acquiring genuine imprecise maps for standard map acquisition datasets (e.g. nuScenes) would be costly and time consuming, **we synthetically generate imprecise existing maps from true HDMaps**.

We develop MapModEX, a standalone map modification library. It takes nuScenes map files and sample records, and for each sample outputs polyline coordinates for dividers, boundaries and pedestrian crosswalks in a given patch, around the ego vehicle. Importantly, our library provides the ability to modify these polylines to reflect various modifications: removal of map elements, addition, shifting of pedestrian crossings, noise addition to point coordinates, map shift, map rotation and map warping. MapModEX will be made available after publication to facilitate further research into incorporating existing maps into online HDMap acquisition from sensors.

We implement **three challenging scenarios**, outlined next, using our MapModEX package, generating for each sample 10 variants of scenarios **2a**, **2b**, **3a** and **3b** (scenario **1** only admits one variant). We chose to work with a fixed set of modified maps to reduce online computation costs during training and to reflect real situations where only a finite number of map variants might be available.

## 3.3. Scenario 1: Maps with only boundaries

A first scenario is one where only a bare HDMap (with road boundaries but without divider or pedestrian crossing) is available as shown on Fig. 2b. Road boundaries are more often associated with 3D physical landmarks (e.g. edge of sidewalk) whereas dividers and pedestrian crossings are generally denoted by flat painted markings that are easier to miss. Moreover, pedestrian crossings and lane dividers are fairly commonly displaced by construction works or road deviations, or even partially hidden by tire tracks.

As such, it is reasonable to use HDMaps with **only road boundaries**. This would have the benefit of reducing annotators costs by only asking annotators to label road boundaries. Furthermore, less precise equipment and less updates might be required to situate only road boundaries.

*Implementation* **Scenario 1** removes the divider and pedestrian crossings from available HDMaps.

### 3.4. Scenarios 2a and 2b: Maps Are Noisy

A second plausible case involves noisy maps as shown on Fig. 2c. A weak point of existing HDMaps is the need for high precision (in the order of a few centimeters), which puts a significant strain on their acquisition and maintenance [12]. In fact, a key difference between HDMaps and the emergent MDMaps standard lies in a lower precision (a few centimeters vs. a few meters).

We therefore propose to work with **noisy HDMaps to simulate a cheaper acquisition process or a shift to the MDMaps standard**. More interestingly, common geolocalization errors or acquiring HDMaps with automatic methods could also lead to noisy maps. Although methods like MapTRv2 have reached very impressive performance, they are not yet completely precise: the Mean Average Precision of predicted maps struggles to reach even 70%.

*Implementation* We propose two possible implementations of these noisy HDMaps to reflect the various conditions under which we might be lacking precision. In a first **Scenario 2a**, we propose a **shift-noise setting** where we add noise from a Gaussian distribution with standard deviation of 1 meter on the localization of each map element. This has the effect of applying a uniform translation to the points defining a given map element (divider, boundary, crosswalk). Such a setting should be a good approximation of situations where human annotators provide quick imprecise annotations from noisy data. We chose a standard deviation of 1 meter to reflect MDMaps standards of being precise up to a few meters [18].

We then test our approach with a very challenging **pointwise-noise setting** in **Scenario 2b**: for each ground truth point - keeping in mind a map element is made up of 20 such points - we sample noise from a Gaussian distribution with standard deviation of 5 meters and add it to the point coordinates. This provides a worst case approximation of a possible situation - in the future - where models automatically acquire maps or where very imprecise localizations are used.

### 3.5. Scenarios 3a and 3b: Maps Have Changed

The final scenario we consider is one where we have access to **old maps that used to be accurate** (see Fig. 2d). As noted in Sec. 3.3, it is fairly common for painted markers like pedestrian crossings to be displaced from time to time. Furthermore, it is not uncommon for cities to substantially remodel some problematic intersection or renovate districts to accommodate traffic increase by a new attraction [41].

It is therefore interesting to use existing HDMaps that are valid on their own but differ from the true HDMaps in significant ways. These maps should often appear when the HDMaps are only updated by the maintainer every few years to cut down on costs. In that case, the available maps would still provide some information on the world but might not reflect temporary or recent changes.

*Implementation* We approximate this situation by applying strong changes to true HDMaps in our **Scenario 3a**. We delete 50% of the pedestrian crossings and lane dividers, add a few pedestrian crossings (half the amount of the remaining crossings) and finally apply a small warping distortion to the map.

However, it is important to note that a substantial amount of the global map will remain unchanged over time. We account for that in our **Scenario 3b**, where we randomly choose (with probability p=0.5) to keep the true HDMap instead of the perturbed existing HDMap.

## 4. MapEX: Accounting for EXisting Maps

In order to verify our central postulate on the usefulness of existing maps, we propose MapEX (see Fig. 3), a novel framework for online HDMap estimation. It follows the classic query based online HDMap estimation framework [32, 35], and two key modules to process existing maps: a **map query encoding** module (see Sec. 4.2) and a **pre-attribution** of predictions to known ground truth for training (see Sec. 4.3). We also discuss an optional change detection module in Appendix Sec. 10. Since our implementation is built upon the state-of-the-art MapTRv2 [33], it will translate to most methods [11, 32, 35]

### 4.1. Overview

**Base framework** The classic query based framework uses a few trainable components (gray on Fig. 3): a sensor to BEV **encoder**, learnable detection **queries** and a map **decoder**. It takes sensor inputs, processes them and outputs predicted map elements (see Appendix Fig. 4).

It starts by taking **sensor inputs** (cameras and/or LiDAR), and **encodes** them into a Bird's Eye View (BEV) representation to serve as sensor features. The map itself is obtained using a **DETR-like** [8] detection scheme to detect the map elements ($N$ at most). It passes $N \times L$ **learned query** tokens ($N$ being the maximum number of detected elements, $L$ the number of points predicted for an element, with $L = 20$ in this paper) into a transformer map **decoder** that feeds sensor information to the query tokens using **cross-attention** with the BEV features. The **decoded queries** are then translated into map element **coordinates** by linear layers along with a class prediction (including an extra background class) such that groups of $L$

queries represent the $L$ points of a map element. **Training** is done by finding a **matching** $\sigma$ between predicted map elements $\{\hat{y}_i = (\hat{c}_i, \hat{p}_i)\}_i$ and true (ground truth) map elements $\{y_i = (c_i, p_i)\}_i$ (possibly padded with empty elements) using some variant of the Hungarian algorithm [9, 25]. Once matched, the model is optimized using a regression loss $\mathcal{L}_{reg}$ (for coordinates) and classification (for element classes) losses $\mathcal{L}_{cls}$:

$$\mathcal{L} = \frac{1}{N} \sum_{i=0}^{N-1} \mathcal{L}_{cls}(\hat{c}_{\sigma(i)}, c_{\sigma(i)}) + \mathcal{L}_{reg}(\hat{p}_{\sigma(i)}, p_{\sigma(i)}). \quad (1)$$

**Our MapEX framework** Classic frameworks do not take existing maps as input, which necessitates new modules at two key levels: at the query level we create non-learnable **EX queries** that complement classic learnable queries (details in Sec. 4.2), and at the matching level we **pre-attribute** predictions to ground truths (details in Sec. 4.3).

The complete MapEX framework - shown on Fig. 3 - creates non-learnable EX queries that **encode** the existing map information. We then complete this set with classic learnable queries to reach a set number of queries $N \times L$. This completed set of queries is then passed to a transformer decoder and translated into predictions by linear layers (as usual). At **training** time, our **attribution module** pre-attributes predictions to known ground truth correspondences before matching, and the rest is matched normally using Hungarian Matching. The **same loss** $\mathcal{L}$ (as in the classic *base framework*) optimizes the **same overall model** (we add no learnable parameters). At **test** time, the decoded non-background queries yield a HDMap representation.

### 4.2. Translating Maps into EX Queries

There is no mechanism in current online HDMap estimation frameworks to account for existing maps. We therefore need to design a new scheme that can translate existing maps into a form understandable by standard query-based online HDMap estimation frameworks. We propose with MapEX a simple method of encoding existing map elements into EX queries for the decoder as shown on Fig. 3.

For a given map element, we extract $L$ evenly spaced points, with $L$ being the number of points we seek to predict for any map element. For each point, we craft an EX query that encodes, in the first 2 dimensions, its **map coordinates (x,y)**, and in the next 3 dimensions a one-hot encoding of the **map element class** (divider, crossing or boundary). The rest of the EX query is padded with 0s to reach the standard query size used by the decoder architecture.

While this query design is very simple, it presents the key benefits of both directly encoding the information of interest (point coordinates and element class), and minimizing collisions with learned queries (thanks to the abundant 0-padding). A detailed discussion is provided in Sec. 5.3 with experimental comparisons to other possible designs.

Once we have $N_{EX}$ sets of $L$ queries (for the $N_{EX}$ map elements in the existing map), we retrieve $(N - N_{EX})$ sets of $L$ assorted learnable queries from our pool of classic learnable queries. The resulting $N \times L$ queries are then fed to the decoder following a base classic method (e.g. VectorMapNet, MapTRv2, ...). After we predict map elements from the queries, we can either directly use them (at test time) or match them to the ground truth for training.

### 4.3. Map Element Pre-attribution for Training

While EX queries introduce a way to account for existing map information, nothing ensures these queries will be properly used by the model to estimate the corresponding elements. In fact, experiments in Sec. 5.3 show the network can fail to identify even fully accurate EX queries, if left on its own. We thus introduce a pre-attribution of predictions to corresponding true map elements before the traditional Hungarian matching used at training as shown on Fig. 3.

Put plainly, **we keep track for each map element in the existing map of which true map element they correspond to**: if a map element is unmodified, shifted or warped we can tie it to the original map element in the true map. To ensure the model learns to solely use useful information, we only keep matches when the average point-wise displacement, between the modified map element $m^{EX} = \{(x_0^{EX}, y_0^{EX}), \ldots, (x_{L-1}^{EX}, y_{L-1}^{EX})\}$ and true map element $m^{GT} = \{(x_0^{GT}, y_0^{GT}), \ldots, (x_{L-1}^{GT}, y_{L-1}^{GT})\}$:

$$s(m^{EX}, m^{GT}) = \left\| \frac{1}{L} \sum_{i=0}^{L-1} \begin{pmatrix} x_i^{EX} \\ y_i^{EX} \end{pmatrix} - \begin{pmatrix} x_i^{GT} \\ y_i^{GT} \end{pmatrix} \right\|_2 \quad (2)$$

is below 1 meter long. In case of deletions or additions, there are no corresponding map elements.

Given the correspondence between ground truth and predicted map elements, **we can then remove the pre-attributed map elements** from the pool of elements to be matched. The remaining map elements (predicted and ground truth) are then matched using some variant of the Hungarian algorithm as per usual [9, 25]. As such, the Hungarian matching step is only needed to identify which EX queries correspond to non-existent added map elements, and to find classic learned queries that fit some of the true map elements absent from the existing map (due to deletion or a strong perturbation).

Reducing how many elements are fed to a Hungarian algorithm is important as even the most efficient variants are of cubical complexity $\mathcal{O}(N^3)$ [9]. This is not a major weak point in online HDMap estimation currently as the predicted maps are small [14,33] ($30m \times 60m$) and only three types of map elements are predicted. As online map generation progresses further however, we will have to accommodate an

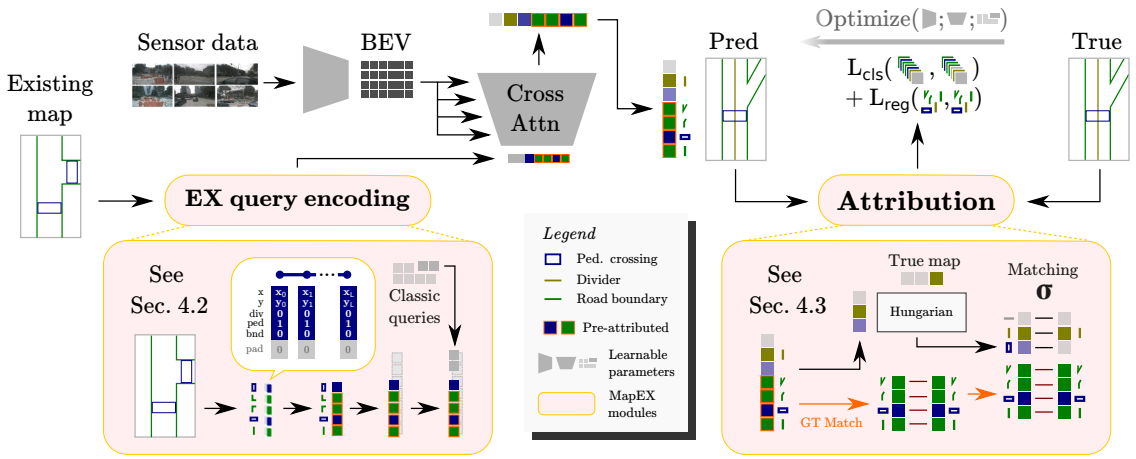



Figure 3. **Overview of our MapEX method** (see Sec. 4). We add two modules (**EX query encoding**, **Attribution**) to the classic query based pipeline. Map elements are encoded into EX queries that are added to classic learned queries. These queries are then decoded using sensor data to yield map elements. Pre-attribution of known prediction to true map elements helps train stronger models.

ever increasing number of map elements as predicted maps grow both larger [14] and more complete [29].

## 5. Experimental Results

We can now verify experimentally our central claim that existing maps are useful for online HDMap estimation by evaluating MapEX (Sec. 4) on reasonable scenarios (Sec. 3). After providing a general comparison of MapEX results in relation to the literature in Sec. 5.1, we highlight the improvements from using existing map information over the baseline in our different scenarios (see Sec. 5.2). We then provide deeper understanding of the MapEX framework through careful ablations in Sec. 5.3.

*Setting* We evaluate our MapEX framework on the **nuScenes dataset** [5] as it is the standard evaluation dataset for online HDMap estimation. We base ourselves on the MapTRv2 framework and official codebase. Following usual practices, we report the **Average Precision** for each of the three map element types (divider, boundary, crossing) at different retrieval thresholds (Chamfer distance of 0.5m, 1.0m and 1.5m) along with overall **mean Average Precision** over the three classes. As these averaged metrics can be difficult to interpret, we provide more granular results in Appendix Sec. 9. To be comparable to results in the literature [11, 33, 35], we show results on the nuScenes val set but conduct no hyper-parameter tuning on the val set to avoid overfitting to it. We directly get training parameters from MapTRv2 without tuning (using standard learning rate scaling heuristics [15] to adapt to our 2 GPU infrastructure). Our code will be made available on the MultiTrans project's official github: https://github.com/anr-multitrans. A complete description of the setting is provided in Appendix Sec. 7.

For each experiment, we conduct **3 runs** using three fixed random seeds. Importantly, for a given seed and scenario combination, the existing map provided during validation is fixed to facilitate comparisons. We report results as $mean \pm std$, up to a decimal point even if standard deviation exceeds that precision, in order to keep notations uniform.

### 5.1. MapEX vs. Other Methods

We provide in Tab. 1 an overview of the literature along with MapEX performance in the 5 existing map scenarios from Sec. 3: maps with no dividers or pedestrian crossings (**S1**), noisy maps (**S2a** for shifted map elements, **S2b** for strong pointwise noise), and substantially changed maps (**S3a** with only those maps, **S3b** with true maps mixed in). Further comparison against important baselines (e.g. **using only the input existing map**) is provided in Tab. 3. We contextualize MapEX's performance by comparing it both to an exhaustive inventory of existing online HDMap estimation on comparable settings (Camera inputs, CNN Backbone) and to the current state-of-the-art (which uses significantly more resources) for the standard map estimation problem. While this leaves out some work [37, 48] on non-standard formulations, this should help contextualize our results.

First, it is clear from Tab. 1 that **any sort of existing map information leads MapEX to significantly outperform the literature** on comparable settings in any scenario. In three scenarios, existing map information even allows MapEX to perform much better than the current state-of-the-art MapNeXt [30] that relies on a powerful foundation model image backbone [47]. Even the fairly conservative **S2a** scenario with imprecise map element localizations leads to an improvement of 6.3 mAP score (i.e. 8%).

**In all scenarios, we observe consistent improvements** over the base MapTRv2 model in all 4 metrics. Understand-

Table 1. **MapEX vs. current methods.** In all possible scenarios, MapEX improves upon the base MapTRv2 model. In Scenarios **2a**, **3a** and **3b** it even beats the state-of-the-art obtained with a much stronger pretrained foundation backbone. Best results from methods are highlighted in **bold**, second best are underlined, third in *italic*. (*: Concurrent work, †: Same codebase and setting as our experiments.)

| Method | Backbone | Epoch | Extra info | Average Precision at {0.5m, 1.0m, 1.5m} | | | |
|---|---|---|---|---|---|---|---|
| | | | | $AP_{divider}$ | $AP_{ped}$ | $AP_{boundary}$ | $mAP$ |
| Previous methods | | | | | | | |
| HDMapNet [28] | EB0 | 30 | ✗ | 27.7 | 10.3 | 45.2 | 27.7 |
| + P-MapNet* [2] | EB0 | 30 | Geoloc. SDMaps | 32.1 | 11.3 | 48.7 | 30.7 |
| VectorMapNet [35] | R50 | 110 | ✗ | 47.3 | 36.1 | 39.3 | 40.9 |
| + Neural Map [49] | R50 | 110 | Learned map feats | 49.6 | 42.9 | 41.6 | 44.8 |
| MapTR [32] | R50 | 24 | ✗ | 51.5 | 46.3 | 53.1 | 50.3 |
| + MapVR [52] | R50 | 24 | ✗ | 54.4 | 47.7 | 51.4 | 51.2 |
| + Satellite Map [14] | R50 | 24 | Geoloc. Satellite views | 55.3 | 47.2 | 55.3 | 52.6 |
| + ADMap [22] | R50 | 24 | ✗ | 56.2 | 49.4 | 57.9 | 54.5 |
| PivotNet [11] | R50 | 24 | ✗ | 56.2 | 56.5 | 60.1 | 57.6 |
| BeMapNet [42] | R50 | 30 | ✗ | 62.3 | 57.7 | 59.4 | 59.8 |
| MapTRv2† [33] | R50 | 24 | ✗ | 62.4 | 59.8 | 62.4 | 61.5 |
| + GeMap* [52] | R50 | 24 | Segmentation loss | 69.8 | 67.1 | 71.4 | 69.4 |
| SQD-MapNet* [46] | R50 | 24 | Prev. frames info | 66.6 | 63.6 | 64.8 | 65.0 |
| MapNeXT* [30] | R50 | 24 | ✗ | 58.8 | 50.3 | 58.7 | 56.0 |
| MapTRv2 [33] | V2-99 | 110 | Depth pretrain | 73.7 | 71.4 | 75.0 | 73.4 |
| MapNeXT* [30] | II-H | 110 | Foundation backbone | 79.3 | *77.4* | 78.8 | 78.5 |
| Only existing map (no sensors) | | | | | | | |
| **S1** maps | R50 | 24 | Map w/ only boundaries | $40.1 \pm 0.4$ | $24.3 \pm 2.4$ | $99.8 \pm 0.1$ | $54.7 \pm 0.9$ |
| **S2a** maps | R50 | 24 | Map w/ element shift | $65.6 \pm 0.8$ | $62.9 \pm 0.6$ | $79.9 \pm 1.1$ | $69.5 \pm 0.5$ |
| **S2b** maps | R50 | 24 | Map w/ point noise | $50.6 \pm 0.5$ | $39.1 \pm 0.3$ | $40.7 \pm 0.6$ | $43.5 \pm 0.4$ |
| **S3a** maps | R50 | 24 | Outdated maps | $65.8 \pm 0.6$ | $41.7 \pm 0.4$ | $98.4 \pm 0.1$ | $73.3 \pm 0.3$ |
| **S3b** maps | R50 | 24 | 50% Outdated maps | $86.0 \pm 0.3$ | $72.5 \pm 0.5$ | $99.2 \pm 0.1$ | $85.9 \pm 0.3$ |
| Our method | | | | | | | |
| MapEX-**S1** | R50 | 24 | Map w/ only boundaries | $66.1 \pm 0.6$ | $62.5 \pm 0.4$ | $\mathbf{99.9 \pm 0.1}$ | $76.2 \pm 0.1$ |
| MapEX-**S2a** | R50 | 24 | Map w/ element shift | $\mathit{82.5 \pm 1.0}$ | $\underline{78.4 \pm 0.8}$ | $93.5 \pm 0.4$ | $\mathit{84.8 \pm 0.3}$ |
| MapEX-**S2b** | R50 | 24 | Map w/ point noise | $78.4 \pm 0.1$ | $62.1 \pm 0.6$ | $72.4 \pm 0.4$ | $70.9 \pm 0.3$ |
| MapEX-**S3a** | R50 | 24 | Outdated maps | $\underline{84.6 \pm 0.3}$ | $74.1 \pm 0.6$ | $\underline{99.1 \pm 0.1}$ | $\underline{85.9 \pm 0.2}$ |
| MapEX-**S3b** | R50 | 24 | 50% outdated maps | $\mathbf{92.8 \pm 0.1}$ | $\mathbf{87.2 \pm 0.1}$ | $\underline{99.3 \pm 0.2}$ | $\mathbf{93.1 \pm 0.1}$ |

ably, **Scenario 3b** (with accurate existing maps half of the time) yields the best overall performance by a large margin, thereby demonstrating a strong ability to recognize and leverage fully accurate existing maps. Both **Scenarios 2a** (with shifted map elements) and **3a** (with "outdated" map elements) offer very strong overall performance with good performance for all three types of map elements. **Scenario 1**, where only road boundaries are available, shows large mAP gains thanks to its (expected) very strong retrieval of boundaries. Even the incredibly challenging **Scenario 2b**, where Gaussian noise of standard deviation 5 meters is applied to each map element point, leads to substantial gains on the base model with particularly good retrieval performance for dividers and boundaries These results are further validated by [3] which builds on our work.

Furthermore, MapEX also **significantly outperforms directly using the input existing map** (with some corrections from a learned model). In all scenarios, results show MapEX substantially improves upon this baseline: its performance cannot solely be attributed to memorizing the map or the information in the existing map.

## 5.2. MapEX Improvements

We now focus more specifically on the improvements that existing map information brings to our base MapTRv2 model. For reference, we compare MapEX gains with those brought by other sources of additional information: **Neural Map Prior** with a global learned feature map [49], **Satellite Maps** with geolocalized Satellite views [14], and **P-MapNet** which uses geolocalized SDMaps [2]. Importantly, MapModEX relies on a stronger base model than these methods. While this makes it harder to improve upon the base model, it also makes it easier to reach high scores. To avoid having an unfair advantage, we provide in Tab. 2 the absolute $\Delta AP = AP^{Base+Info} - AP^{Base}$ score gain (which is the standard metric observed in [2, 14, 49]).

We see from Tab. 2 that using **any kind of existing map with MapEX leads to overall mAP gains larger than using any other source** of additional information (including a more sophisticated P-MapNet setting). We generally observe very strong improvements to the model's detection performance on both lane dividers and road boundaries. A slight exception is **Scenario 1** (where we only have access to road boundaries) where the model successfully retains map information on boundaries but only provides improvements comparable to previous methods on the two map elements it has no prior information on. Pedestrian crossings seem to require more precise information from existing maps as both **Scenario 1** and **Scenario 2b** (where a very destructive noise is applied to each map point) only pro-

Table 2. **Improvements from additional information.** In all considered scenarios, existing map information substantially improves results compared to other sources of information. ($^*$: Concurrent work, $^+$: MAE [20] pretraining, Camera+LiDAR inputs)

| Method | Improvement $\Delta AP = AP^{Base+Info} - AP^{Base}$ | | | |
|---|---|---|---|---|
| | $\Delta AP_{divider}$ | $\Delta AP_{ped}$ | $\Delta AP_{bound}$ | $\Delta mAP$ |
| Previous methods | | | | |
| Neural Map | +02.3 | +06.8 | +02.6 | +03.9 |
| Satellite Map$^*$ | +03.8 | +00.9 | +02.2 | +02.3 |
| P-MapNet$^*$ | +04.4 | +01.0 | +03.5 | +03.0 |
| P-MapNet$^{*,+}$ | +08.4 | +11.1 | +06.8 | +08.8 |
| Our method | | | | |
| MapEX-S1-onlybounds | +03.7 | +02.8 | +37.5 | +14.7 |
| MapEX-S2a-shift-noise | +20.1 | +18.6 | +31.1 | +23.3 |
| MapEX-S2b-point-noise | +16.0 | +02.3 | +10.0 | +09.4 |
| MapEX-S3a-fullchange | +22.2 | +14.3 | +36.7 | +21.4 |
| MapEX-S3b-halfchange | +30.4 | +27.4 | +36.9 | +31.6 |

vide improvements comparable to existing techniques. **Scenarios 2a** (with shifted elements) and **3a** (with "outdated" maps) lead to strong detection scores for pedestrian crossings, which might be because these two scenarios contain more precise information on pedestrian crossings.

### 5.3. Ablations on the MapEX Framework

Table 3. **Influence of MapEX pre-attribution** on mAP.

| Method | Mean Average Precision | | | | |
|---|---|---|---|---|---|
| | S1 | S2a-shift | S2b-noise | S3a-full | S3b-half |
| MapEX | **76.2** $\pm$ **0.1** | **84.8** $\pm$ **0.3** | **70.9** $\pm$ **0.3** | **85.9** $\pm$ **0.2** | **93.1** $\pm$ **0.1** |
| w/o Pre-Attribution | 64.5 $\pm$ 1.9 | **84.7** $\pm$ **0.7** | **72.0** $\pm$ **1.9** | 80.3 $\pm$ 9.6 | **93.1** $\pm$ **0.2** |

**Contribution of inputs in MapEX** Tab. 3 shows ground truth correspondences (for pre-attribution of predictions and ground truths) seem to lower the variance of MapEX as indicated by lines 1 and 2 of Tab. 3. This demonstrates that **pre-attribution is indeed necessary to properly leverage existing map information**. A good way to understand this is to consider our **Scenario 1**. In this scenario, we have access to the exact boundary elements. With pre-attribution this consistently leads to near perfect retrieval of those elements (see Tab. 1). This is not the case without pre-attribution unfortunately: in two out of three runs, the network only reaches a score below 80% AP. This suggests pre-attribution helps ensure MapEX consistently learns to utilize the information provided by existing maps.

Table 4. **Influence of map queries** (**Scenario 1**). Our non-learnable EX query perform well while requiring no extra training.

| Method | Average Precision at {0.5m, 1.0m, 1.5m} | | | |
|---|---|---|---|---|
| | $AP_{divider}$ | $AP_{ped}$ | $AP_{boundary}$ | $mAP$ |
| MapEX encoding | 66.1 $\pm$ 0.6 | 62.5 $\pm$ 0.4 | 99.9 $\pm$ 0.1 | 76.2 $\pm$ 0.1 |
| Linear encoding | 63.4 $\pm$ 0.3 | 61.1 $\pm$ 0.3 | 100 $\pm$ 0.1 | 74.8 $\pm$ 0.1 |
| Lin. enc. w/ MapEx init. | 66.6 $\pm$ 0.1 | 62.5 $\pm$ 0.9 | 100 $\pm$ 0.1 | 76.4 $\pm$ 0.3 |

**On EX query encoding** We use a simple encoding to translate existing map elements into EX queries. One might expect learned EX queries - in line with concurrent work on map encoding [37, 46] - to be more useful (e.g. by projecting a 5-dimensional vector description into a query). However, Tab. 4 shows learned EX queries perform much worse than ours. Interestingly, initializing learnable EX query with the non-learnable values might bring very minor improvements that do not justify the added complexity.

Table 5. **Influence of pre-attribution threshold** (**Scenario 2a**).

| Method | Average Precision at {0.5m, 1.0m, 1.5m} | | | |
|---|---|---|---|---|
| | $AP_{divider}$ | $AP_{ped}$ | $AP_{boundary}$ | $mAP$ |
| MapEX | 82.5 $\pm$ 1.0 | 78.4 $\pm$ 0.8 | 93.5 $\pm$ 0.4 | 84.8 $\pm$ 0.3 |
| ... w/o sim. thresh. | 79.5 $\pm$ 1.6 | 76.4 $\pm$ 0.9 | 91.9 $\pm$ 0.2 | 82.6 $\pm$ 0.7 |

**On ground truth pre-attribution threshold** Since pre-attributing map elements is important to consistently use existing map information (see Tab. 3), it might be tempting to pre-attribute all the corresponding map elements instead of filtering them. Tab. 5 shows that discarding correspondences when the existing map element is too different (see Sec. 4.3) does lead to stronger performance than indiscriminate attribution. In essence, it is preferable to use a learnable query instead of EX queries when the existing map element is too different from the ground truth.

## 6. Discussion

We improve online HDMap estimation with an overlooked resource: **existing maps**. We outline three **realistic scenarios** where existing (minimalist, noisy or outdated) maps are available. As current frameworks cannot use existing maps, we develop two novel MapEX modules: one encoding map elements into **EX queries**, and another that **pre-attributes** predictions to known ground truth correspondences to ensure the model leverages these queries.

Experimental results demonstrate that existing maps represent a crucial information for online HDMap estimation, with MapEX significantly improving upon comparable methods regardless of the scenario. In fact, the median scenario (in terms of mAP) - **Scenario 2a** with randomly shifted map elements - improves upon the base MapTRv2 model by **38%** and upon the current state-of-the-art by **8%**.

We hope this work will lead new online HDMap estimations to account for existing information. Existing maps - good or bad - are **widely available**. To ignore them is to forego a **crucial** tool in reliable online HDMap estimation.

*Acknowledgements* This work was realized by the MultiTrans project funded by the Agence Nationale de la Recherche under grant reference ANR-21-CE23-0032. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

## References


[1] Learning to drive like a human. https://wayve.ai/thinking/learning-to-drive-like-a-human/ (2019), 2019-04-03 1, 2

[2] Anonymous: P-mapnet: Far-seeing map constructer enhanced by both SDMap and HDMap priors. In: Submitted to ICLR (2023), under review 2, 3, 7

[3] Bateman, S.M., Xu, N., Zhao, H.C., Ben Shalom, Y., Gong, V., Long, G., Maddern, W.: Exploring real world map change generalization of prior-informed hd map prediction models. In: CVPR workshop (2024) 2, 7

[4] Bu, T., Mertz, C., Dolan, J.: Toward map updates with crosswalk change detection using a monocular bus camera. In: IEEE Intelligent Vehicles Symposium (2023) 3

[5] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. arXiv preprint (2019) 2, 6

[6] Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Structured bird's-eye-view traffic scene understanding from onboard images. In: ICCV (2021) 1, 2

[7] Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Topology preserving local road network estimation from single onboard camera image. In: CVPR (2022) 2, 3

[8] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020) 4

[9] Crouse, D.F.: On implementing 2d rectangular assignment algorithms. IEEE Transactions on Aerospace and Electronic Systems pp. 1679–1696 (2016) 5

[10] Deo, N., Wolff, E., Beijbom, O.: Multimodal trajectory prediction conditioned on lane-graph traversals. In: CoRL (2021) 2

[11] Ding, W., Qiao, L., Qiu, X., Zhang, C.: Pivotnet: Vectorized pivot learning for end-to-end hd map construction. In: ICCV (2023) 3, 4, 6, 7, 2

[12] Elghazaly, G., Frank, R., Harvey, S., Safko, S.: High-definition maps: Comprehensive survey, challenges, and future perspectives. IEEE Open Journal of Intelligent Transportation Systems (2023) 1, 2, 3, 4

[13] Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation. In: CVPR (2020) 1

[14] Gao, W., Fu, J., Jing, H., Zheng, N.: Complementing onboard sensors with satellite map: A new perspective for hD map construction. ICRA (2024) 2, 3, 5, 6, 7

[15] Granziol, D., Zohren, S., Roberts, S.: Learning rates as a function of batch size: A random matrix theory approach to neural network training. The Journal of Machine Learning Research (2022) 6, 1

[16] Gu, J., Hu, C., Zhang, T., Chen, X., Wang, Y., Wang, Y., Zhao, H.: Vip3d: End-to-end visual trajectory prediction via 3d agent queries. In: CVPR (2023) 1

[17] Gulzar, M., Muhammad, Y., Muhammad, N.: A survey on motion prediction of pedestrians and vehicles for autonomous driving. IEEE Access (2021) 2

[18] Gupta, R.: The mapping singularity is near. https://medium.com/@ro_gupta/the-mapping-singularity-is-near-85dc4577b33d (2021), 2021-04-08 2, 4

[19] Hausler, S., Garg, S., Chakravarty, P., Shrivastava, S., Vora, A., Milford, M.: Displacing objects: Improving dynamic vehicle detection via visual place recognition under adverse conditions. IROS (2023) 1, 2

[20] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022) 8

[21] Heo, M., Kim, J., Kim, S.: HD map change detection with cross-domain deep metric learning. In: IROS (2020) 3

[22] Hu, H., Wang, F., Wang, Y., Hu, L., Xu, J., Zhang, Z.: Admap: Anti-disturbance framework for reconstructing online vectorized hd map. arXiv preprint (2024) 7

[23] Jeong, J., Yoon, J.Y., Lee, H., Darweesh, H., Sung, W.: Tutorial on high-definition map generation for automated driving in urban environments. Sensors (2022) 1, 2

[24] Kim, K., Cho, S., Chung, W.: HD map update for autonomous driving with crowdsourced data. IEEE Robotics and Automation Letters (2021) 3

[25] Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly (1955) 5

[26] Kuutti, S., Bowden, R., Jin, Y., Barber, P., Fallah, S.: A survey of deep learning applications to autonomous vehicle control. IEEE Transactions on Intelligent Transportation Systems (2020) 1

[27] Lambert, J.W., Hays, J.: Trust, but Verify: Cross-modality fusion for hd map change detection. In: NeurIPS Datasets and Benchmarks (2021) 2, 3, 1

[28] Li, Q., Wang, Y., Wang, Y., Zhao, H.: Hdmapnet: An online hD map construction and evaluation framework. In: ICRA (2022) 2, 3, 7

[29] Li, T., Chen, L., Geng, X., Wang, H., Li, Y., Liu, Z., Jiang, S., Wang, Y., Xu, H., Xu, C., et al.: Topology reasoning for driving scenes. arXiv preprint (2023) 3, 6

[30] Li, T.: Mapnext: Revisiting training and scaling practices for online vectorized hd map construction. arXiv preprint (2024) 2, 6, 7

[31] Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: ICCV (2020) 1

[32] Liao, B., Chen, S., Wang, X., Cheng, T., Zhang, Q., Liu, W., Huang, C.: MapTR: Structured modeling and learning for online vectorized HD map construction. In: ICLR (2023) 1, 3, 4, 7

[33] Liao, B., Chen, S., Zhang, Y., Jiang, B., Zhang, Q., Liu, W., Huang, C., Wang, X.: Maptrv2: An end-to-end framework for online vectorized hd map construction. arXiv preprint (2023) 1, 2, 3, 4, 5, 6, 7

[34] Liu, M., Cheng, H., Chen, L., Broszio, H., Li, J., Zhao, R., Sester, M., Yang, M.Y.: LAformer: Trajectory Prediction for Autonomous Driving with Lane-Aware Scene Constraints. arXiv preprint (2023) 2

[35] Liu, Y., Yuantian, Y., Wang, Y., Wang, Y., Zhao, H.: Vectormapnet: End-to-end vectorized hd map learning. In: ICML (2023) 1, 2, 3, 4, 6, 7

[36] Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal motion prediction with stacked transformers. CVPR (2021) 2

[37] Luo, K.Z., Weng, X., Wang, Y., Wu, S., Li, J., Weinberger, K.Q., Wang, Y., Pavone, M.: Augmenting lane perception and topology understanding with standard definition navigation maps. arXiv preprint (2023) 2, 6, 8

[38] Pannen, D., Liebner, M., Burgard, W.: HD map change detection with a boosted particle filter. In: ICRA (2019) 3

[39] Pannen, D., Liebner, M., Hempel, W., Burgard, W.: How to keep hD maps for automated driving up to date. In: ICRA (2020) 3

[40] Park, D., Ryu, H., Yang, Y., Cho, J., Kim, J., Yoon, K.J.: Leveraging future relationship reasoning for vehicle trajectory prediction (frm). In: ICLR (2023) 1, 2

[41] Plachetka, C., Maier, N., Fricke, J., Termöhlen, J.A., Fingscheidt, T.: Terminology and analysis of map deviations in urban domains: Towards dependability for hd maps in automated vehicles. In: IEEE Intelligent Vehicles Symposium (2020) 4

[42] Qiao, L., Ding, W., Qiu, X., Zhang, C.: End-to-end vectorized hD-map construction with piecewise bezier curve. In: CVPR (2023) 3, 7

[43] Sun, R., Lingrand, D., Precioso, F.: Exploring the road graph in trajectory forecasting for autonomous driving. In: ICCV workshops (2023) 2

[44] Teng, S., Hu, X., Deng, P., Li, B., Li, Y., Ai, Y., Yang, D., Li, L., Xuanyuan, Z., Zhu, F., Chen, L.: Motion planning for autonomous driving: The state of the art and future perspectives. IEEE Transactions on Intelligent Vehicles (2023) 1

[45] Wang, H., Li, T., Li, Y., Chen, L., Sima, C., Liu, Z., Wang, B., Jia, P., Wang, Y., Jiang, S., Wen, F., Xu, H., Luo, P., Yan, J., Zhang, W., Li, H.: Openlane-v2: A topology reasoning benchmark for unified 3d hD mapping. In: NeurIPS (2023) 3

[46] Wang, S., Jia, F., Liu, Y., Zhao, Y., Chen, Z., Wang, T., Zhang, C., Zhang, X., Zhao, F.: Stream query denoising for vectorized hd map construction. arXiv preprint (2024) 7, 8

[47] Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., et al.: Internimage: Exploring large-scale vision foundation models with deformable convolutions. In: CVPR (2023) 2, 6

[48] Wu, X., Lau, K., Ferroni, F., Ošep, A., Ramanan, D.: Pix2map: Cross-modal retrieval for inferring street maps from images. In: CVPR (2023) 2, 6

[49] Xiong, X., Liu, Y., Yuan, T., Wang, Y., Wang, Y., Zhao, H.: Neural map prior for autonomous driving. In: CVPR (2023) 3, 7

[50] Xu, Y., Chambon, L., Éloi Zablocki, Chen, M., Alahi, A., Cord, M., Pérez, P.: Towards motion forecasting with real-world perception inputs: Are end-to-end approaches competitive? In: ICRA (2024) 2

[51] Xu, Z., Liu, Y., Sun, Y., Liu, M., Wang, L.: Centerlinedet: Road lane centerline graph detection with vehicle-mounted sensors by transformer for high-definition map creation. ICRA (2023) 2

[52] Zhang, Z., Zhang, Y., Ding, X., Jin, F., Yue, X.: Online vectorized hd map construction using geometry. arXiv preprint (2023) 3, 7

## Supplementary material

We provide in this Appendix some additional details to understand our work:

- We provide more details on our experimental setting in Sec. 7.
- We discuss how the Argoverse 2 Trust but Verify relates to our problem in Sec. 8.
- We provide the detailed precision tables and qualitative examples for our main results in Sec. 9.
- We study how the model behaves with exact map inputs in Sec. 10.
- We give pseudocode overviews of our two original MapEX modules in Sec. 11.
- We give a figure of a query based onlline HDMap estimation framework without MapEX modules in Fig. 4.

## 7. Detailed setting and codebase

We introduce here the detailed experimental details used for our experiments along with in-depth explaination of how existing maps are obtained for our various scenarios. Our code is largely based on the official MapTRv2 code[1], and will be made available along with our standalone MapModEX library on the MultiTrans project's official github: https://github.com/anr-multitrans.

**Training details** We largely reprise the 24 epochs training settings from our MapTRv2 [33] base, which were described in the original paper as:

> "ResNet50 is used as the image back- bone network unless otherwise specified. The optimizer is AdamW with weight decay 0.01. The batch size is 32 (containing 6 view images) and all models are trained with 8 NVIDIA GeForce RTX 3090 GPUs. Default training schedule is 24 epochs and the initial learning rate is set to $6 \times 10$-4 with cosine decay. We extract ground-truth map elements in the perception range of ego-vehicle following [...] The resolution of source nuScenes images is $1600 \times 900$. [...] Color jitter is used by default in both nuScenes dataset and Argoverse2 dataset. The default number of instance queries, point queries and decoder layers is 50, 20 and 6, respectively. For PV-to-BEV transformation, we set the size of each BEV grid to 0.3m and utilize efficient BEVPoolv2 [77] operation. Following [16], $\lambda_c = 2$, $\lambda_p = 5$, $\lambda_d = 0.005$. For dense prediction loss, we set $\alpha_d, \alpha_p, \alpha_b$ to 3, 2 and 1 respectively. For the overall loss, $\beta_o = 1$, $\beta_m = 1$, $\beta_d = 1$."

Our own training setting solely differs from MapTRv2's in the fact that we train on 2 NVIDIA Quadro RTX 8000 GPUs. This in turn mean we need to reduce the batch size by 4 and scale learning rates by 2 following standard scaling heuristics for Adam optimizers [15].

**Scenario 1 implementation** We remove the divider and pedestrian crossings from available HDMaps.

**Scenario 2a implementation** For each map element localization, we add noise from a Gaussian distribution with standard deviation of 1 meter. This has the effect of applying a uniform translation to each map element (dividers, boundaries, crosswalks).

**Scenario 2b implementation** For each ground truth point - keeping in mind a map element is made up of 20 such points - we sample noise from a Gaussian distribution with standard deviation of 5 meters and add it to the point coordinates.

**Scenario 3a implementation** We delete 50% of the pedestrian crossings and lane dividers in the map, add a few pedestrian crossings (half the amount of the remaining crossings) and finally apply a small warping distortion to the map. The warping distortion is composed of first trigonometric warping with horizontal and vertical amplitudes 1, and inclination 3. We then perform triangular warping following a slightly perturbed grid where each point on the regular grid is shifted according to random Gaussian noise with standard deviation 1.

**Scenario 3b implementation** For each map, we draw a uniform random value between 0 and 1. If it is below p=0.5 we keep the true HDMap, otherwise we perturb it in the same way as in Scenario 3a.

## 8. On the Trust but Verify dataset

The Argoverse 2 Trust but Verify (TbV) dataset [27] offers situations where the HDMap does not fit sensor inputs for change detection. Unfortunately, it is **not suitable** for our purposes it only says whether the current map fits sensor data (yes or no) **without giving the new map** (see Sec. 3.3 of [27] or the associated code). Without the relevant ground truth we cannot evaluate on it.

[1] https://github.com/hustvl/MapTR/tree/maptrv2

Additionally, while TbV is an excellent dataset for change detection, it unfortunately contains a limited number of real scenarios to train model for online HDMap acquisition. Moreover, a number of the change scenarios are indiscernible for our HDMap representation (e.g. change in the type of divider). Interestingly, the limited number of hand curated change situations is reserved for the validation and test sets with the train set generated from synthetic data. Where TbV chooses to generate synthetic views that differ from the available HDMap, we take the opposite view of modifying the HDMaps. While this is likely less desirable for change detection, it is of no consequence for online HDMap acquisition and much lighter computationally.

# 9. Fine grained results of map estimations

Tab. 6 provides a deeper look into the detailed results of MapEX and sheds light on how the different types of existing maps actually benefit the model.

Interestingly, the noisy **Scenarios 2a** and **2b** seem to help the model give a rough approximation of map elements (good scores for retrieval thresholds of 1.5m) but are less useful when it comes to predict very precise element localizations. As such, these scenarios appear to help the model by providing a **general idea of what the situation looks like**. Nevertheless, **Scenarios 2a** appears to still subtantially improve the base MapTRv2 model for precise element localizations at 0.5m (which is much lower than the standard deviation of the added noise).

Conversely, when outdated map **Scenarios 3a** and **3b** are useful to predict map elements, they tend to provide fairly precise element localizations (the gap between precision at 0.5m and 1.5m is lower). While these scenarios strongly improve performance at all precision thresholds, the improvement is also much larger for very precise element localizations. As such, they seem to work by providing a **more precise approximations of map elements**.

**Scenario 1** (with only boundaries) for its part shines by providing near perfect estimations of map boundaries at all levels: it properly **identifies the provided road boundary localizations as perfectly accurate** and restitutes them as is. Interestingly, it also provides significant gains in precision at all retrieval thresholds for lane dividers and pedestrian crossings even though the existing map has no information on them.

# 10. Map change detection

We discuss here an additional module initially explored for MapEX. We include this discussion here as this module does not improve performance (and is therefore not an improved version of MapEX), but sheds light on what happens when perfect existing maps are available to the model.

## 10.1. Map change detector

There are a number of situations where fully accurate HDMaps might be mixed in with the imperfect HDMaps (e.g. our **Scenario 3b**). As such, we propose a lightweight change detection module to leverage these situations.

We introduce a learned change detection query token and perform cross-attention between this token and intermediate map element queries at different stages of the decoder. This token is then decoded by dense layer into a change prediction $c \in [0, 1]$ (with a sigmoid activation). At training time, we train this token with a binary cross entropy loss (with target $\hat{c} = 1$ if the map is not fully accurate and $\hat{c} = 0$ if it is): we minimize

$$\mathcal{L} = \mathcal{L}_{Base} + \mathcal{L}_{BCE}(c, \hat{c}), \tag{3}$$

with $\mathcal{L}_{Base}$ the loss of the base online HDMap estimator. At test time, if no change is detected we output the existing HDMap instead of the prediction (and we output the decoder predictions as usual if a change is detected).

Using the existing HDMap has two benefits: it provides a very precise HDMap (something most methods struggle with [11]), and it provides a way to stop the map estimation process early. Indeed, returning the existing map removes the need for further decoding of the query tokens which can be expensive.

## 10.2. Processing accurate existing maps

We take a closer look at how MapEX deals with perfectly accurate existing maps as it can sometimes happen in scenarios like **Scenario 3b**. To this end, we compare MapEX to variants that use an explicit map change detection module (described in Appendix 10) and substitute the predicted map with the input existing map if no change is detected. Tab. 7 shows MapEX does not need a change detection module: it recognizes and uses accurate existing map elements on its own. In fact, training a change detection module jointly with MapEX appears to deteriorate performance.

Table 7. Usefulness of the change detection module (**Scenario 3b**). MapEX seems to recognize and leverage existing maps without the need for explicit change detection.

| Method | Average Precision at {0.5m, 1.0m, 1.5m} | | | |
|---|---|---|---|---|
| | $AP_{divider}$ | $AP_{ped}$ | $AP_{boundary}$ | $mAP$ |
| MapEX | $92.8 \pm 0.1$ | $87.2 \pm 0.1$ | $99.3 \pm 0.2$ | $93.1 \pm 0.1$ |
| ... w/ substitution | $92.5 \pm 0.3$ | $87.3 \pm 0.3$ | $99.4 \pm 0.1$ | $93.0 \pm 0.1$ |
| ... w/ sub. & optimization | $92.5 \pm 0.2$ | $87.2 \pm 0.2$ | $99.3 \pm 0.1$ | $93.0 \pm 0.1$ |

# 11. Pseudo code

We provide here pseudo code for our two additional modules: the EX query encoding module (Alg. 1) and the pre-attribution code (Alg. 2).

Table 6. Detailed table of retrieval Precisions at different thresholds for the main results. Reproduced scores for the base MapTRv2 model are given for reference.

(a)

| Method | $AP_{divider}$ | | |
|---|---|---|---|
| | $Precision^{0.5}_{divider}$ | $Precision^{1.0}_{divider}$ | $Precision^{1.5}_{divider}$ |
| MapTRv2 | 46.0 | 66.4 | 75.4 |
| MapEX-S1 | $50.7 \pm 0.3$ | $69.6 \pm 0.7$ | $77.8 \pm 0.6$ |
| MapEX-S2a | *62.8 ± 2.1* | *83.6 ± 1.4* | $92.1 \pm 1.3$ |
| MapEX-S2b | $50.9 \pm 3.0$ | $77.5 \pm 2.2$ | $89.0 \pm 1.0$ |
| MapEX-S3a | $76.2 \pm 0.5$ | $86.6 \pm 0.3$ | *90.8 ± 0.3* |
| MapEX-S3b | **88.4 ± 0.5** | **93.8 ± 0.4** | **95.8 ± 0.2** |

(b)

| Method | $AP_{ped}$ | | |
|---|---|---|---|
| | $Precision^{0.5}_{ped}$ | $Precision^{1.0}_{ped}$ | $Precision^{1.5}_{ped}$ |
| MapTRv2 | 34.5 | 65.1 | 78.8 |
| MapEX-S1 | $38.8 \pm 0.5$ | $68.8 \pm 0.5$ | $80.0 \pm 0.5$ |
| MapEX-S2a | *46.5* $\pm 0.5$ | $85.5 \pm 1.9$ | **97.2 ± 0.4** |
| MapEX-S2b | $28.4 \pm 0.5$ | $72.3 \pm 2.1$ | *91.7 ± 1.8* |
| MapEX-S3a | $56.2 \pm 0.4$ | *79.4 ± 0.5* | $86.7 \pm 0.7$ |
| MapEX-S3b | **77.7 ± 0.5** | **89.9 ± 0.3** | $93.6 \pm 0.3$ |

(c)

| Method | $AP_{boundary}$ | | |
|---|---|---|---|
| | $Precision^{0.5}_{boundary}$ | $Precision^{1.0}_{boundary}$ | $Precision^{1.5}_{boundary}$ |
| MapTRv2 | 39.6 | 70.3 | 80.6 |
| MapEX-S1 | **99.8 ± 0.1** | **99.8 ± 0.1** | **100.0 ± 0.1** |
| MapEX-S2a | $80.5 \pm 0.9$ | $96.4 \pm 0.4$ | $98.9 \pm 0.2$ |
| MapEX-S2b | $34.9 \pm 0.2$ | $75.8 \pm 0.2$ | $90.0 \pm 0.3$ |
| MapEX-S3a | $97.4 \pm 0.3$ | **99.9 ± 0.1** | **100.0 ± 0.1** |
| MapEX-S3b | $97.8 \pm 0.4$ | **99.7 ± 0.3** | **100.0 ± 0.1** |

**Data:** Map element $m^{EX} = \{(x_0^{EX}, y_0^{EX}), \ldots, (x_{L-1}^{EX}, y_{L-1}^{EX})\}$ of class $c$ (among divider, crossing and boundary).
**Result:** list query_list of $L$ $H$-dimensional EX queries.
query_list = [];
**for** $i \leftarrow 0$ **to** *L-1* **do**
  /* Encode position */
  pos_vec = array($[x_i^{EX}, y_i^{EX}]$);
  /* Encode class */
  class_vec = one_hot($c$, num_class=3);
  /* Build query */
  pad_vec = zeros($H - 5$);
  query_i = concatenate([pos_vec, class_vec, pad_vec]);
  query_list.append(query_i);
**end**
**return** query_list;

**Algorithm 1:** Encoding map elements into EX Queries.

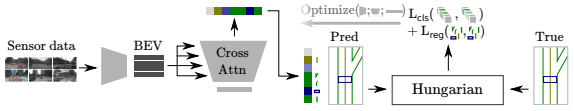



Figure 4. **Overview of a classic query based framework**. Sensor data is encoded into BEV features, before being cross attended with learned detection queries in a DETR-like scheme. The final attended queries serve to predict coordinates and classes of map elements. The model is trained using a Hungarian matching between predictions and ground truths.

**Data:** Predictions $p = \{p_i\}_{i=0,\ldots,49}$, (Padded) ground truths $g = \{g_i\}_{i=0,\ldots,49}$, correspondence list $c = \{c_i\}_{i=0,\ldots,49}$ where $c_i = -1$ if there is no correspondence
**Result:** Assignment $a = \{a_i\}_{i=0,\ldots,49}$ where $a_i$ is the index of the ground truth associated to the $i$-th prediction.
/* Split off pre-attributed pairs */
$p^p, g^p, c^p, p^n, g^n, c^n, split\_inds$ = Split(p,g,c);
/* Perform Hungarian matching */
$a^n$ = Hungarian($p^n, g^n$);
/* Merge $c^p$ with $a^n$ */
a = Merge($p^p$, $a^n$, split_inds); **return** a;

**Algorithm 2:** Hungarian matching with pre-attribution.