Pix2Map: Learning to regress street maps from images

Xindi Wu[1], KwunFung Lau[1], Francesco Ferroni[2], Deva Ramanan[1,2]
[1]Robotics Institute, Carnegie Mellon University, [2]Argo AI
{xindiw, kwunfunl, deva}@cs.cmu.edu, fferroni@argo.ai

Abstract

In this paper, we propose Pix2Map, a method for regressing urban street maps from ego-view images. Such technology is a core component of many current self-driving fleets, both for offline map construction and online map maintenance (to accommodate potential changes in street layout due to construction, etc.). This problem is challenging because street maps are often represented as discrete graphs with varying number of lane nodes and edge connectivities, while sensory inputs are continuous arrays of images. We advocate a retrieval perspective, and contrastively learn multimodal embeddings for both ego-view images and graph-structured street maps. We then use these embeddings at test-time to perform cross-modal retrieval; given an ego-view image and its embedding, return the street map (from a training library) with the most similar embedding. We analyze the performance of our method using various graph similarity metrics motivated by urban planning (such as connectivity, density, etc.), and show compelling performance on the Argoverse dataset.

1. Introduction

Most autonomous robots make use of prior information about the scene, often encoded as a map [28]. As an illustrative example, almost all autonomous vehicle stacks make use of maps that encode geometric and semantic information about the surrounding scene, such as lane-level boundaries and the locations of signs. Maps can serve as an enormously powerful *prior* about the surrounding scene. When integrated with on-the-fly sensor measurements from a camera (or lidar), they can provide an accurate view of the surrounding world. However, such a map prior can be difficult to build, as map construction is notoriously labor intensive (often requiring manual effort [20,27]). Moreover, the prior can be detrimental when not up-to-date, due to dynamic urban environments that are often in flux due to construction, turbulent weather conditions, etc [15]. This motivates the challenge of automatic map updates and more broadly the need for dynamic up-to-date maps.

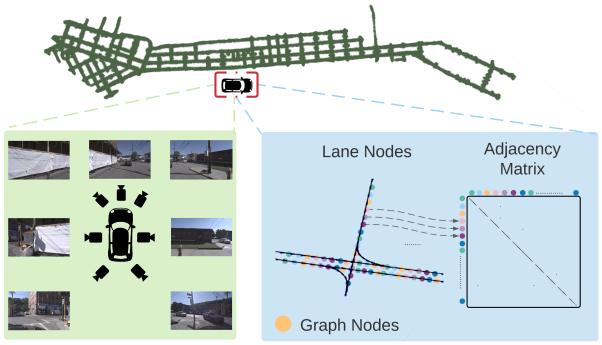

Figure 1. Illustration of the camera images (left), vector-map (right) and the global street map (top) to represent street view and high-definition map. The adjacency matrix is shown next to a spatial visualization of the nodes and their edges.

Pix2Map. In this work, we advocate a simple formulation of the map *construction* and map *maintenance* problem as a visual Pix2Map task: given the ego-view images, generate street maps. Given the recent multi-city autonomous vehicle datasets [5], it is straightforward to construct pairs of ego-view images and street view maps, both for training and testing. By training and testing on image-graph pairs from different geographic locations (e.g., train on Pittsburgh, test on Miami), one can simulate the map-construction problem. By training and testing on the *same* geographic location (e.g., train and test on Pittsburgh), one can simulate the map-maintenance problem.

Why is it hard? Ego-view sensors are continuous images, but street maps are discrete graphs with varying number of nodes and topology (depending on the particular road layout) , requiring a different set of priors [2]. This makes it hard to formalize the problem as a (nonlinear) regression task, that simply takes pixels as input and returns graphs. Initial approaches for graph generation tend to make use of iterative *decoders* that generate nodes and their connectivity in a sequential [34] or hierarchical manner [22], but graph generation remains very much an open problem.

Cross-modal retrieval. Instead, our core insight is to sidestep the problem of graph generation by recasting

Pix2Map as a cross-modal retrieval task: given a set of test-time ego-view images, we (1) compute its visual embedding and then (2) retrieve a graph with the closest graph embedding in terms of cosine similarity. Crucially, we jointly learn image and graph encoders that are trained to operate in the same embedding space, making use of recent techniques for cross-modal contrastive learning (ie., CLIP [24]). In order to do so, our key technical contribution is a novel but simple graph *encoder* for extracting fixed-dimensional embeddings from street graphs of arbitrary size and topology, adapting sequential transformer models from the language community for this purpose (i.e., BERT [9]).

Experimental analysis. We demonstrate the efficacy of cross-modal retrieval for the Pix2Map problem on real-world data from Argoverse [5]. Interestingly, the training data (of image and graph pairs) used to learn the encoders may be different from the graph library used for retrieval. We demonstrate better accuracy with larger libraries, even with fixed encoders. In the supplement, we also show pilot experiments for "Map2Pix", which retrieves a close-matching image (from an image library) given a graph. While not the focus of our work, such technology might be useful for generating simulated worlds for validation [22].

To summarize, the main contributions include:

- We propose a contrastive cross-modal approach to dynamic street map construction from camera data. Specifically, we train a graph encoder and image encoder with a shared latent space building on recent advances in multimodal representation learning.

- We define a new task and benchmark for map maintenance and map update, evaluating both fidelity and generalization. We leverage the Argoverse [5] dataset to define a comprehensive and rigorous set of evaluation criteria over these tasks.

- Empirically, we demonstrate that this approach is effective in this new domain and perform ablation studies to highlight the impacts of architectural decisions. We demonstrate that this approach has the ability to generalize, both to novel observations within a city as well as to unseen cities.

2. Related Work

Map generation. Maps in autonomous vehicles have multiple, potentially co-existing representations. Grid maps, voxel maps or semantic point clouds, are commonly used "low-level" data structures [4, 10, 21, 25, 26, 33]. However, of equal interest are more compact entity-based data structures like vector maps and spatial graphs. Spatial graphs are generic, as they can represent geometric structures like lanes, intersections, their attributes, connectivity information and so on. Many state-of-the-art trajectory generation algorithms use graph representations of maps as priors, so being able to retrieve/generate graphs directly from sensory data becomes of interest.

Most works in vector map generation literature [19, 30] focus specifically on lane detection in images/LIDAR and frame it as an object detection task, attempting to localise lanes and then predict their shapes. However, this covers only a fraction of potential map layouts (i.e. it ignores intersections). There are multiple datasets for lane detection [1, 8, 32, 35]. Other methods have several output representations (semantic masks, instance masks, direction masks) that are then transformed into a directed spatial graph as a non-learnt post-processing step [17].

Somewhat tangentially, recent works have also applied deep learning to graph embedding and generation. One representation attempts to build a detailed 3D geometric mesh of one's surroundings [26], relying on lidar and semantic segmentation techniques for a high-resolution semantically labeled representation. [29] created a large benchmark which contains different categories of image sources for various purposes including semantic map generation. Many others are designed simply to generate random graphs from a distribution resembling that of a target graph dataset, but not to reconstruct particular graphs conditioned on sensor data [11, 22, 34].

Contrastive learning. Contrastive learning is a collection of machine learning approaches which aim to leverage relationships between examples that samples matching and non-matching samples to efficiently learn a useful underlying feature space [16]. Contrastive learning is closely related to metric learning, where a model attempts to learn an embedding space that learns to bring similar examples closer together and separate unrelated examples [31] and has been demonstrated as an effective technique for few-shot visual learning [18]. There is a large body of work proposing visual contrastive learning techniques including the widely-used SimCLR [6] and MoCo [7, 13].

More recently, with impressive results from [24] on a collection of vision and language tasks such as zero-shot classification, optical character recognition, and action classification, contrastive learning has been shown as an effective approach for learning a shared latent space across multiple modalities. This is accomplished by learning different encoders for each modality trained so that paired examples are close together while the others are far apart.

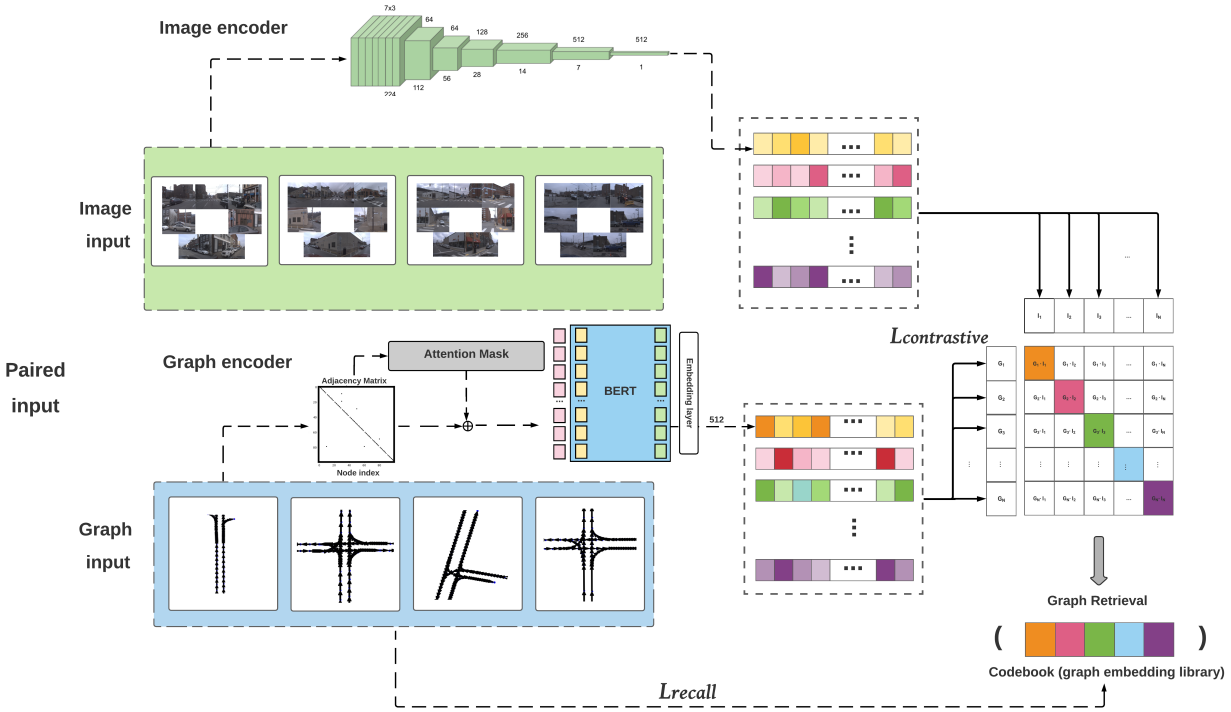

Figure 2. Graph encoder and image encoder provided the graph embedding and image embedding separately, and then those vector representations are used to build a similarity matrix, which contrasts the image and graph embeddings. We highlight that the adjacency matrix of a given graph is used as the attention mask of the BERT-based transformer model operating on that graph.

3. Method

In this section, we will formulate the map maintenance and update the task as a graph retrieval and reconstruction problem. We will present our proposed method Pix2Map for jointly training the image and graph encoder to find the shared embedding space, and further infer vector maps from images. We will first present our problem setting and notation, following a detailed description of our method.

3.1. Problem formulation

Assume we have a training library of image-graph pairs, written as (I, G) where I is a list of 7 ego-view images from a camera ring and G is an street map represented as a graph $G = (V, E)$, where each vertex $v \in V$ represents a lane *node* and $E \in \{0, 1\}^{|V|\times|V|}$ represents the connectivity between all lane nodes (adjacency matrix). Importantly, lane nodes are represented with an $v = (x, y)$ position in a local egocentric "birds-eye-view" coordinate frame (such that (0,0) is the ego-vehicle location). Importantly, we allow for different graphs G to have different numbers of lane nodes and connectivity.

Note that the Argoverse dataset by default describes the relationship between lane segments, and treats the nodes which compose a lane segment as a property of that lane segment. However, since we operate at the node level we need to preprocess the segment-level data into node-level data. We treat sequential nodes within a segment as connected, with directed edges according to the lane direction. In addition, we use the successor lane property of a lane segment to create an edge from the final node of a lane segment to the first node of any lane segment that succeeds it.

We use our training library of (image,graph) pairs to learn both an image encoder and graph encoder, as detailed in the following sections.

3.2. Image encoder

Given an image I, we learn an encoder that produces a fixed-dimensional embedding

$$\phi_{\text{image}}(I) \in R^{512}$$

We use a standard image encoder, ResNet18 [14] as a feature extractor by removing its fully connected layer. We experiment with "from-scratch" training as well as with models using ImageNet-pre-trained weights. In order to process the n input images (we use $n = 7$ images throughout our work), we stack them channelwise and then in the pre-trained case replace the first convolution layer with one that stacks the original pre-trained filters n times, with each weight divided by n.

Compared to encode the seven images separately and merge the features, ideally if the model can learn to understand the relationships between the input images, it could better understand what the street layout is. We reuse the original weights when making the new convolutional filters so the benefits of the pretrained weights will be preserved.

3.3. Graph encoder

Given a graph $G = (V, E)$, we would like to produce a fixed dimensional embedding:

$$\phi_{\text{graph}}(G) \in R^{512}$$

Unlike pixels in an image or words in a sentence, nodes in graphs do not have an inherent order. Hence we would like our graph encoder to be invariant to orderings of graph nodes. To do so, we construct a graph encoder using a transformer architecture defined on lane-graph nodes, inspired by sequence-to-sequence architectures from language [9] that process sequences containing tokens. We treat lane nodes as a collection of tokens and edges as masks for attentional processing.

$$v \in R^{2k} \tag{1}$$

$$Transformer(v) = \sum_{w \in \text{Neighbor}(v)} \psi_v(v)\text{Softmax}_w \psi_q(v)\psi_k(w) \tag{2}$$

We encode each input vertex $v \in V$ as $2k$-dimensional input embedding obtained by rasterizing its k polyline coordinates. These embeddings are fed into a transformer that computes new embeddings by taking an attentional-weighted average of embeddings from adjacent lane segments. Importantly, we make use of the adjacency matrix E to mask the attentional weighting. We apply $M = 7$ transformer layers, structured similarly to BERT but without positional embeddings. We finally average (or mean pool) all output embeddings to produce a final fixed-dimensional embedding for graph G, regardless of the number of lane segments or their connectivity. In the supplement, we ablate various design choices for our graph encoder.

3.4. Image-graph contrastive learning

We use the cross-modal contrastive formalism of [24], but describe it here for completeness. Given N image-graph pairs (I, G) within a batch, our model jointly learns the encoders $\phi_{\text{image}}(\cdot)$ and $\phi_{\text{graph}}(\cdot)$ such that the cosine similarity of the N correct image-graph pairs will be high and and the $N^2 - N$ incorrect pairs will be low, making use of a symmetric cross entropy loss. Let us write the cosine similarity between an image and graph encoding as follows:

$$\langle x, g \rangle = \frac{x^\top g}{\|x\|\|g\|}, \quad \text{where} \quad x = \phi_{\text{image}}(I), g = \phi_{\text{graph}}(G). \tag{3}$$

We then compute bidirectional contrastive losses composed of an image-to-graph loss $\ell^{(x \to g)}$ and graph-to-image loss $\ell^{(g \to x)}$ which has the same form as the InfoNCE loss [23]:

$$\ell_i^{(x \to g)} = -\log \frac{\exp\langle x_i, g_i \rangle}{\sum_{k=1}^{N} \exp\langle x_i, g_k \rangle} \tag{4}$$

$$\ell_i^{(g \to x)} = -\log \frac{\exp\langle g_i, x_i \rangle}{\sum_{k=1}^{N} \exp\langle g_i, x_k \rangle}. \tag{5}$$

Our final training loss is then computed as a weighted combination of the two losses averaged over all positive image-graph pairs in each minibatch:

$$\ell_{contrastive} = \frac{1}{2N} \sum_{i=1}^{N} \left(\ell_i^{(x \to g)} + \ell_i^{(g \to x)} \right). \tag{6}$$

The above penalizes all incorrect image-graph pairs equally. We found it beneficial to (slightly) prefer incorrect matches with similar graphs, as measured using graph metrics (described in Sec 4.1). Intuitively, we measure the similarity of graph topologies after aligning vertices. Formally, given a ground truth graph G_1 and candidate match G_2, first establish a correspondence between each vertex $v_1 \in V_1$ and its closest match $v_2 \in V_2$ (in terms of euclidean distances of lane centroids). Given such corresponding vertices, compute a binary cross-entropy loss between adjacency matrix E_1 and the permuted matrix E_2.

$$\ell = \ell_{contrastive} + \ell_{edge}. \tag{7}$$

3.5. Pix2Map via retrieval

Given the learned encodings above, we now use them for regressing maps from pixel image input via retrieval. Denoting a graph library as $\mathbb{G}$:

$$I \Rightarrow G^* \quad \text{where} \quad G^* = \underset{G \in \mathbb{G}_{\text{retrieval}}}{\text{argmax}} \langle \phi_{\text{image}}(I), \phi_{\text{graph}}(G) \rangle. \tag{8}$$

Note that the graph library used for retrieval does not need to be the same as the one on which the model was trained. More generally, we note a training dataset $\mathbb{D}_{\text{train}}$ consists of a paired image library, $\mathbb{I}_{\text{train}}$ and graph library, $\mathbb{G}_{\text{train}}$, such that $\mathbb{D}_{\text{train}} := (G_i, I_j) \in \mathbb{G}_{\text{train}} \times \mathbb{I}_{\text{train}}$ where $i = j$. However, we can perform retrieval on any $\mathbb{G}$. Not only does $\mathbb{G}_{\text{retrieval}}$ not need to be equivalent to $\mathbb{G}_{\text{train}}$, but we also do not require there to be a $\mathbb{I}_{\text{retrieval}}$ corresponding to $\mathbb{G}_{\text{retrieval}}$.

Table 1. Number of images and graphs pairs for each city.

City	Train	Validation	Test
Pittsburgh	5701	2544	2170
Miami	7421	2471	1998
Total	**13122**	**5015**	**4168**

Table 2. Number of unique lanes for each city in training, validation and test set. The number of overlapped lanes of the training set and test set are provided. See Fig. 3 for reference.

City	Train	Validation	Test	Overlap
Pittsburgh	480	315	294	145
Miami	1432	443	394	216
Total	**1912**	**758**	**688**	**361**

This has several useful properties. First, we can retrieve graphs from a larger database of graphs than used to train the model or retrieve graphs from a particular subset of the training data such as a city neighborhood. In addition, we can retrieve graphs from cities without corresponding images in the training set or from past maps for which images may not be available. Correspondingly, if we would like to retrieve street map images using a graph (i.e. Map2Pix), as shown in the appendix, $\mathbb{I}_{\text{retrieval}}$ does not need to be equivalent to $\mathbb{I}_{\text{train}}$ and $\mathbb{I}_{\text{retrieval}}$ does not need an associated $\mathbb{G}_{\text{retrieval}}$.

4. Experiments

In this section, we describe the experiments conducted to analyze and evaluate our Pix2Map. To better understand each component's contribution in our framework, we evaluate the graph encoder, image encoder separately and further evaluate the contrastive component.

Dataset We use seven ring camera images including ring front center/left/right, ring rear left/right, ring side left/right. Furthermore, one core feature of Argoverse is that it contains vector maps to describe the connectivity and properties of the lanes. We can base on the provided connectivity to construct various small vector submaps and paired RGB images for training. This is the major reason that we prefer Argoverse than nuScenes [3]. We perform the training and validation of our model on the Argoverse dataset across two cities in the United States, including Pittsburgh(total size of 86 kilometers) and Miami(total size of 204 kilometers of lanes). The image data were captured from seven high-resolution ring cameras (1920×1200) recorded at 30 Hz with overlapping fields of view, providing 360° coverage. We do not use 2 additional front-facing stereo cameras since the resolution is different (2056×2464) and sampled at different frequency (5 Hz).

Map preprocessing The key component of HD maps is the central line of drivable lanes. We include the lane nodes on the lanes within a 40×40 window which is under the reasonable distance. Compared with the map preprocessing method in [17] paper, which throw away 70% of the nodes due to the limitation of graph neural network, we keep 50% of the nodes. We use the adjacency matrix to represent which nodes of a graph are adjacent to which other nodes. We also make sure to rotate the node position and lanes to align the driving direction.

Implementation We train on a single A100 GPU, up to batches of size 256. We train for 40 epochs, where a single epoch takes 40 minutes of wall-clock time. We make use of the Adam optimizer with a learning rate of $2e^{-4}$. Please see supplement for more architecture and training details.

4.1. Metrics

To quantitatively evaluate the quality of the retrieved graphs, we need metrics that capture the difference between the retrieved graph $G_1 = (V_1, E_1)$ and the ground truth $G_2 = (V_2, E_2)$. We closely follow the metrics introduced in HDMapGen, a recent work for street map generation (*not* conditioned on image input) [22].

Spatial point discrepancy We first introduce metrics that represent lane graph nodes $v \in V$ as (x, y) points corresponding the lane centroid, ignoring edge connectivity. We can use metrics for measuring differences between point sets. We first use *Chamfer Distance*: for every $v_1 \in V_1$, compute the closest point in $v_2 \in V_2$ and add all the distances. To ensure a symmetric distance, we similarly add closet-matching distances for all points in $v_2 \in V_2$. We also make use of *Maximum Mean Discrepancy (MMD)* [12], which measures the squared distance between point centroids in a (reproducing) Hilbert space using Gaussian kernels $\langle \varphi(v_1), \varphi(v_2) \rangle_H = k(x_1 - x_2, y_1 - y_2)$:

$$MMD(G_1, G_2) = || \frac{1}{|V_1|} \sum_{v_1 \in V_1} \varphi(v_1) - \frac{1}{|V_2|} \sum_{v_2 \in V_2} \varphi(v_2) ||_H^2$$

Urban Planning We also compute a set of metrics motivated by the urban planning literature. *Urban Connectivity* measures the average degree of a graph node. *Urban Density* captures the density of nodes and edges in particular spatial regions. *Urban Reach* is defined as the total distance covered by traffic lanes, which is designed to capture the development of the urban city. We sum up the length of each graph edge, computed as the distance between node centroids. For the above urban planning metrics, we quantify error as the fraction predicted minus the ground truth, relative to groundTruth.

Geometry fidelity Finally, we use metrics that factor into account the geometric layout of lanes, both in terms of their *length* and *orientation* statistics. We then compute the mean μ and standard deviation σ of lane length and orientation, computing the Fréchet Distance between these statistics:

$$d(G_1, G_2) = (\mu_{G_1} - \mu_{G_2})^2 + (\sigma_{G_1} - \sigma_{G_2})^2 \quad (9)$$

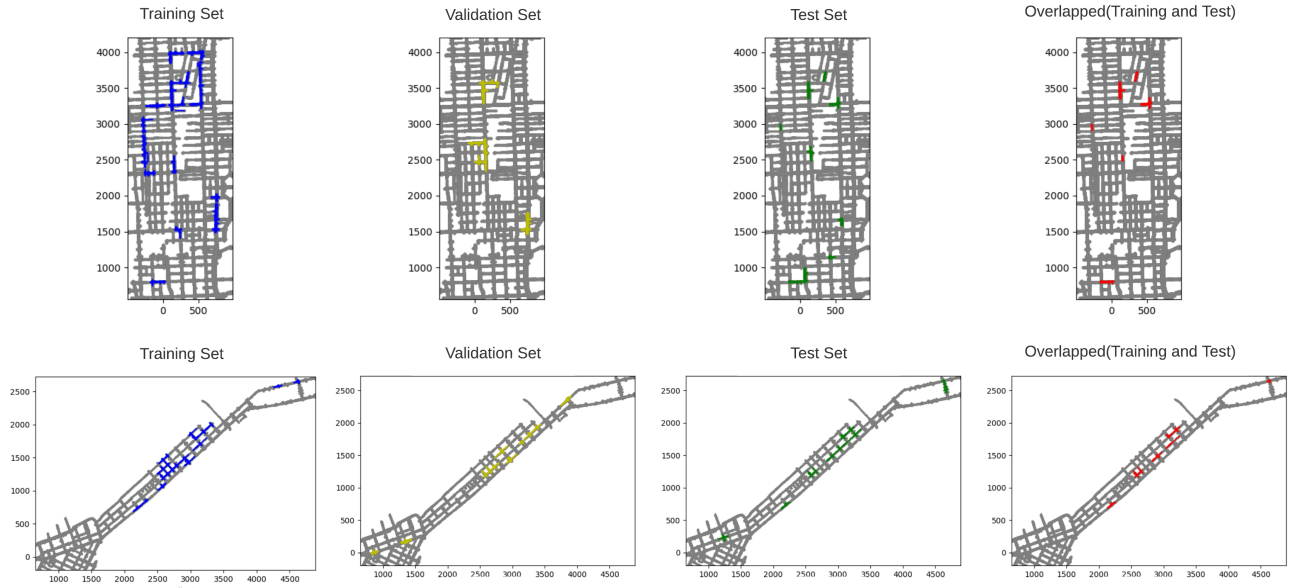

Figure 3. Top: Miami global map, Bottom: Pittsburgh global map. Distribution of trajectories covered in the training/validation/test set and their overlap, **blue**: lanes that appear in training set, **yellow**: lanes that appear in validation set, **red**: lanes that appear both in training set and validation set, **grey**: lanes that have no corresponding ego-view images and position information in the argoverse tracking dataset.

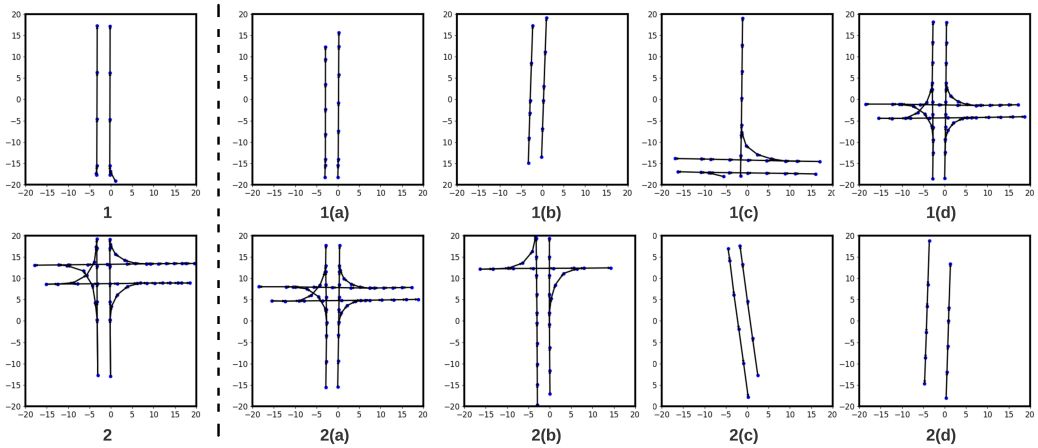

Figure 4. Graph examples are provided to help understand the meaning of the evaluation metric results in Table 3. For most metrics, the query graphs (1 and 2) are considered most similar to (a), and more similar to (b) than to (c). While the Chamfer distance of the query graphs to (a) or (b) is similar, the metric reflects this substantial difference of candidate graphs (c).

4.2. Analysis

Qualitative: We show qualitative retrieval results in Fig. 5. Please see the caption for a detailed description, but generally speaking, we find Pix2Map returns reasonable graphs similar to the ground truth.

Ablations: We include several ablations in our system design to uncover important design decisions. We find that using a pretrained image encoder helps Table **??**. Table 4 summaries the performance of our Pix2Map model with various number of batchsize, it suggests that large batch size can improve the performance effectively and is beneficial to contrastive learning methods. We find that the adjacency-based attention mask improves the performance of the model, as indicated by Table 6. Table 5 suggests that the smaller window size achieves better results, which is not surprising since less graph information is covered

Table 3. Qualitative analysis of evaluation metrics. We compute the metric distance between query graphs (1 and 2) and several candidate matching graphs (1(a),1(b),1(c),1(d)) and (2(a),2(b),2(c),2(d)), as shown in Fig. 4. Qualitatively graphs (a) and (b) match better with the queries. This is corroborated by the metric distances below.

Query	Sample	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1	1a	0.6651	45.9437	0.8132	1.9999	1.6790	2.0000	1.0475
	1b	4.9475	11.8780	0.7356	0	1.6410	0	1.8196
	1c	15.8394	66.7715	24.2401	6.2500	4.0345	6.2500	5.5748
	1d	8.9496	102.6168	17.3458	8.1538	6.2729	8.1538	5.6664
2	2a	2.7565	8.8729	0.0766	0.3174	1.5217	0.3174	1.4290
	2b	2.2447	3.2026	7.0007	0.1389	5.9611	0.8333	1.6700
	2c	10.6054	30.4845	20.8556	9.9998	24.2884	0.5000	8.2707
	2d	14.1146	19.0658	27.5834	9.0909	24.7433	0.4545	7.1300

Table 4. Evaluation of the relationship between batch size on model performance. In line with prior work on contrastive learning and metric learning, we find that using larger batch sizes significantly improves performance.

#	Batchsize	Contrastive 10^{-2}	Recall 10^{-2}	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1	64	10.6973	9.7109	11.6671	10.3307	37.5099	3.6924	**3.4967**	3.6924	4.8802
2	128	6.3714	5.1646	8.9968	70.2874	14.2414	2.6740	3.8975	2.5688	3.2962
3	256	**3.7814**	**2.5075**	**8.9561**	**7.1442**	**1.6704**	**2.69492**	4.3084	**2.2561**	**1.7978**

Table 5. Ablation study to demonstrate the performance impact of using different window size. We see that certain cities (Miami) tend to have larger visibility, as shown by the stronger performance of our model with larger window sizes. Perhaps unsurprisingly, larger window sizes appear to make the task more difficult based on most metrics, with the most notable exception of recall. The impact on recall may be because smaller graphs have less information available to tell them apart.

#	Window Size	Contrastive 10^{-2}	Recall 10^{-2}	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1	30 $\times$30	3.4143	2.8637	6.1953	5.8739	1.3566	1.4235	4.3017	1.2521	1.7835
2	40 $\times$40	3.7814	2.5075	8.9561	7.1442	1.6704	2.69492	4.3084	2.2561	1.7978
3	50 $\times$50	3.8235	2.9345	8.32825	2.8046	1.3566	1.5063	4.3736	2.3972	2.6970

Table 6. Ablation study on the graph encoder model to demonstrate how the performance varies related to whether applying the edge-based attention mask to graph encoder or not, thus specify which tokens (lane node) are attended to and which are not.

#	Attention mask	Contrastive 10^{-2}	Recall 10^{-2}	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1	$\times$	3.8509	2.9421	**4.1643**	9.0245	19.3086	**1.9727**	4.3084	**2.1633**	2.8104
2	✓	**3.7814**	**2.5075**	8.9561	**7.1442**	**1.6704**	2.69492	4.3084	2.2561	**1.7978**

Table 7. Evaluation performance on map maintenance (training and evaluation on the same city) v.s. map construction (training and evaluation on difference cities). One interesting property is that the quality of generalization to different cities is not symmetric: the model trained on Pittsburgh performs better on both the contrastive loss and recall evaluation in Miami when compared to the model trained in Miami and evaluated in Pittsburgh.

#	Test city	Train city	Contrastive 10^{-2}	Recall 10^{-2}	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1	PIT	PIT	**3.7814**	**2.5075**	**8.9561**	**7.1442**	**1.6704**	**2.69492**	4.3084	**2.2561**	**1.7978**
2		MIA	6.4128	2.9435	12.1904	11.641	2.32221	21.1950	**4.3081**	4.4899	3.1656
3	MIA	PIT	6.7440	4.6023	12.3515	8.0938	**6.9270**	5.1031	6.6901	5.0319	4.8690
4		MIA	**3.8681**	**2.9435**	**7.8395**	**4.5742**	3.2598	**3.9575**	**4.1764**	**3.9482**	**2.5782**

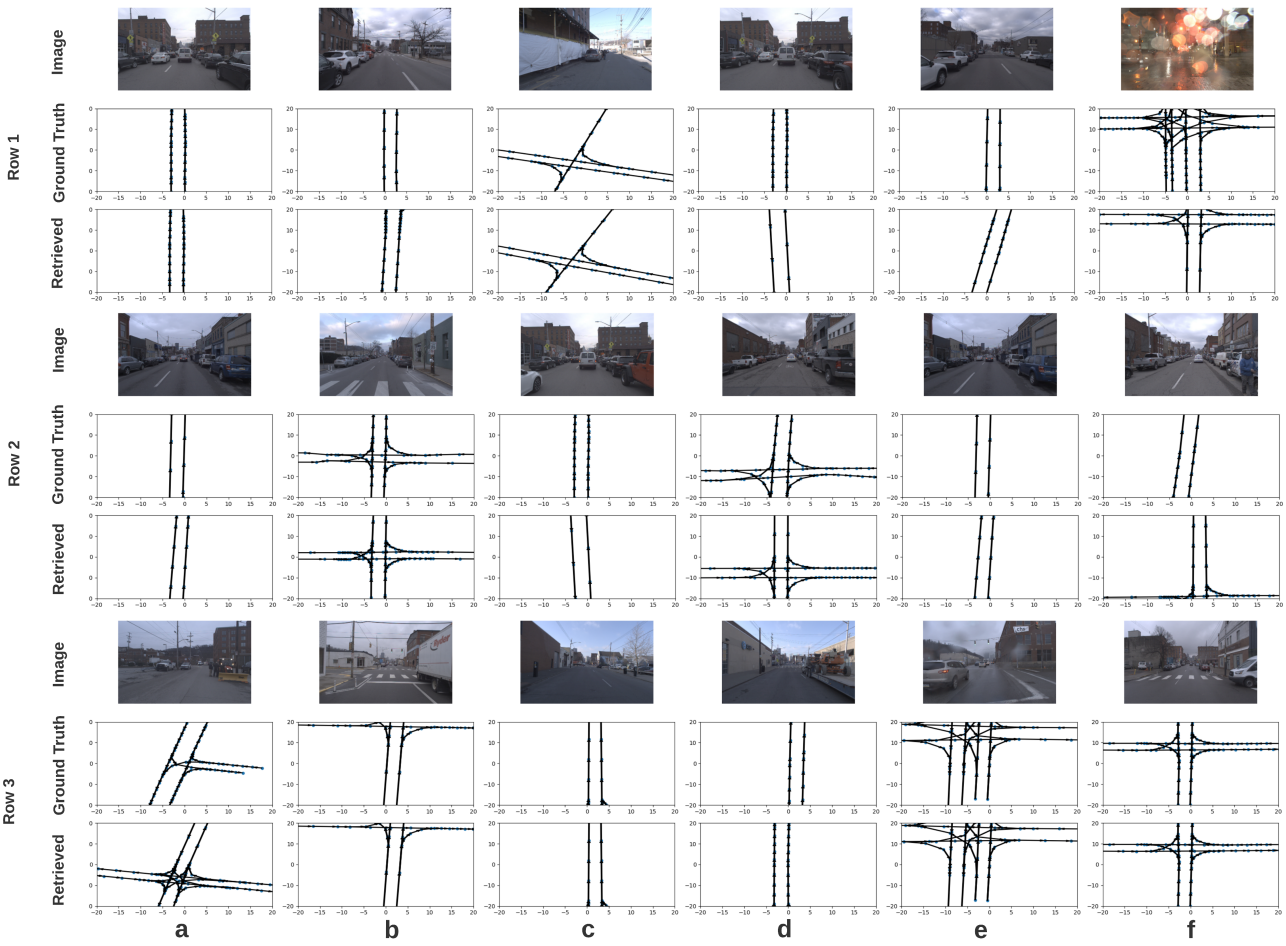

Figure 5. Given a set of ego-view images from the same frame (we show only the front camera for brevity) (**top**), we plot the ground-truth graph (**middle**), followed by Pix2Map prediction obtained via cross-modal retrieval. This graph show results for the map maintenance task (train and test on the same city) and include results for map construction (train and test on different cities) in the supplement. The retrieve result shown in **3e** indicates our model is robust towards certain amount perturbation. As we can see, our method has limited performance when heavy rain occlusion occurs (**1f**) and it become fairly hard to recognize the left/right lane layout under visually degraded condition even for human vision. In general, we find Pix2Map retrieval produces qualitatively similar graphs, compared to the ground truth.

within the test window range. Finally, Table 7 show that map-maintenance (training/testing on the same city) is an easier problem than map-construction (training/testing on different cities). We also include additional ablations (such as increasing the size of the training set, as well as initial experiments for Map2Pix) in the supplement.

5. Conclusion

Overall, our experiments suggest that learning a multi-modal embedding space for camera data and map data is a promising direction, and we hope our work can become an essential building block for map maintenance in the autonomous driving field. However, while these results are encouraging and demonstrate the effectiveness of a constructive approach, there are also many potentially impactful future directions possible. For example, instead of performing graph retrieval, one could use the latent space to generate new unseen graphs using a graph-based decoder architecture. Alternative task-specific encoder architectures are also a direction which may be impactful. The integration of other modalities such as lidar, radar or audio is also a valuable research direction. Even the use of video rather than a set of images at one time, while architecturally similar, may allow the model to leverage depth information and disregard temporary occlusions. Generally, we believe that advances on this task have a high likelihood of being impactful not only practically but also more broadly within the autonomous driving community.

References

[1] Karsten Behrendt and Ryan Soussan. Unsupervised labeled lane markers using maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 5

[4] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semanticmaps and representations from egocentric views. *arXiv preprint arXiv:2010.01191*, 2, 2020. 2

[5] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 1, 2

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2

[8] Wensheng* Cheng, Hao* Luo, Wen Yang, Lei Yu, Shoushun Chen, and Wei Li. Det: A high-resolution dvs dataset for lane extraction. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, 2019. 2

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 4

[10] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019. 2

[11] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017. 2

[12] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. In *Advances in neural information processing systems*, pages 10701–10711, 2019. 5

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[15] John Lambert and James Hays. Trust, but verify: Cross-modality fusion for hd map change detection. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 1

[16] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 2020. 2

[17] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. *arXiv preprint arXiv:2107.06307*, 2021. 2, 5

[18] Xiaomeng Li, Lequan Yu, Chi-Wing Fu, Meng Fang, and Pheng-Ann Heng. Revisiting metric learning for few-shot image classification. *Neurocomputing*, 406:49–58, 2020. 2

[19] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution. *arXiv preprint arXiv:2105.05003*, 2021. 2

[20] Rong Liu, Jinling Wang, and Bingqi Zhang. High definition map for automated driving: Overview and analysis. *The Journal of Navigation*, 73(2):324–341, 2020. 1

[21] Wei-Chiu Ma, Ignacio Tartavull, Ioan Andrei Bârsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. Exploiting sparse semantic hd maps for self-driving vehicle localization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5304–5311. IEEE, 2019. 2

[22] Lu Mi, Hang Zhao, Charlie Nash, Xiaohan Jin, Jiyang Gao, Chen Sun, Cordelia Schmid, Nir Shavit, Yuning Chai, and Dragomir Anguelov. Hdmapgen: A hierarchical graph generative model of high definition maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4227–4236, 2021. 1, 2, 5

[23] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 2, 4

[25] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11138–11147, 2020. 2

[26] Radu Alexandru Rosu, Jan Quenzel, and Sven Behnke. Semi-supervised semantic mapping through label propagation with semantic texture meshes. *International Journal of Computer Vision*, 128(5):1220–1238, 2020. 2

[27] Heiko G Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2):159–162, 2016. 1

[28] Sebastian Thrun. Robotic mapping: a survey. 2003. 1

[29] Shenlong Wang, Min Bai, Gellert Mattyus, Hang Chu, Wenjie Luo, Bin Yang, Justin Liang, Joel Cheverie, Sanja Fidler, and Raquel Urtasun. Torontocity: Seeing the world with a million eyes. *arXiv preprint arXiv:1612.00423*, 2016. 2

[30] Dong Wu, Manwen Liao, Weitian Zhang, and Xinggang Wang. Yolop: You only look once for panoptic driving perception. *arXiv preprint arXiv:2108.11250*, 2021. 2

[31] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:521–528, 2002. 2

[32] Ping Luo Xiaogang Wang Xingang Pan, Jianping Shi and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *AAAI Conference on Artificial Intelligence (AAAI)*, February 2018. 2

[33] Weixiang Yang, Qi Li, Wenxi Liu, Yuanlong Yu, Yuexin Ma, Shengfeng He, and Jia Pan. Projecting your view attentively: Monocular road scene layout estimation via cross-view transformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15536–15545, 2021. 2

[34] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018. 1, 2

[35] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020. 2

Supplementary material for Pix2Map: Learning to regress street maps from images

A. Qualitative Analysis of Map Expansion

In order to qualitatively show the ability for our method to perform map expansion within a city, we train Pix2Map on two distinct subsets of Miami provided by the Argoverse dataset, MIA-1 and MIA-2 and demonstrate these results in Figure 1.

B. Augmenting the Graph Library During Test

While the Argoverse dataset may have not have camera data at every point in a given city, the dataset-provided HD map includes many regions without associated camera information. We can select graph examples from previously unseen regions of the city to drastically expand our retrieval dataset. We visualize these results in Table 1.

C. Ablation Study

Graph Encoder We consider two primary variations on the graph encoder architecture: first, whether we include the adjacency-based attention mask or alternatively allow attention between all nodes passed to the encoder; second, whether we pass in a given row of the adjacency matrix as input to the model for each node alongside its other feature information. We include these results in Table 2. The broadly best-performing method, using an adjacency-based attention mask but not providing the adjacency matrix rows to the model, is the one that we ultimately use in Pix2Map as shown in the main paper.

Loss Design As an ablation on the loss function, we find that removing the recall loss as a training objective worsens the performance of Pix2Map as shown in Table 3.

D. Map2Pix Illustration

It is also possible to retrieve egocentric camera data using a street graphs, which may be useful for simulation. We provide two image retrieval examples in Figure 2, visualizing the front views of $K = 5$ images for each street graph.

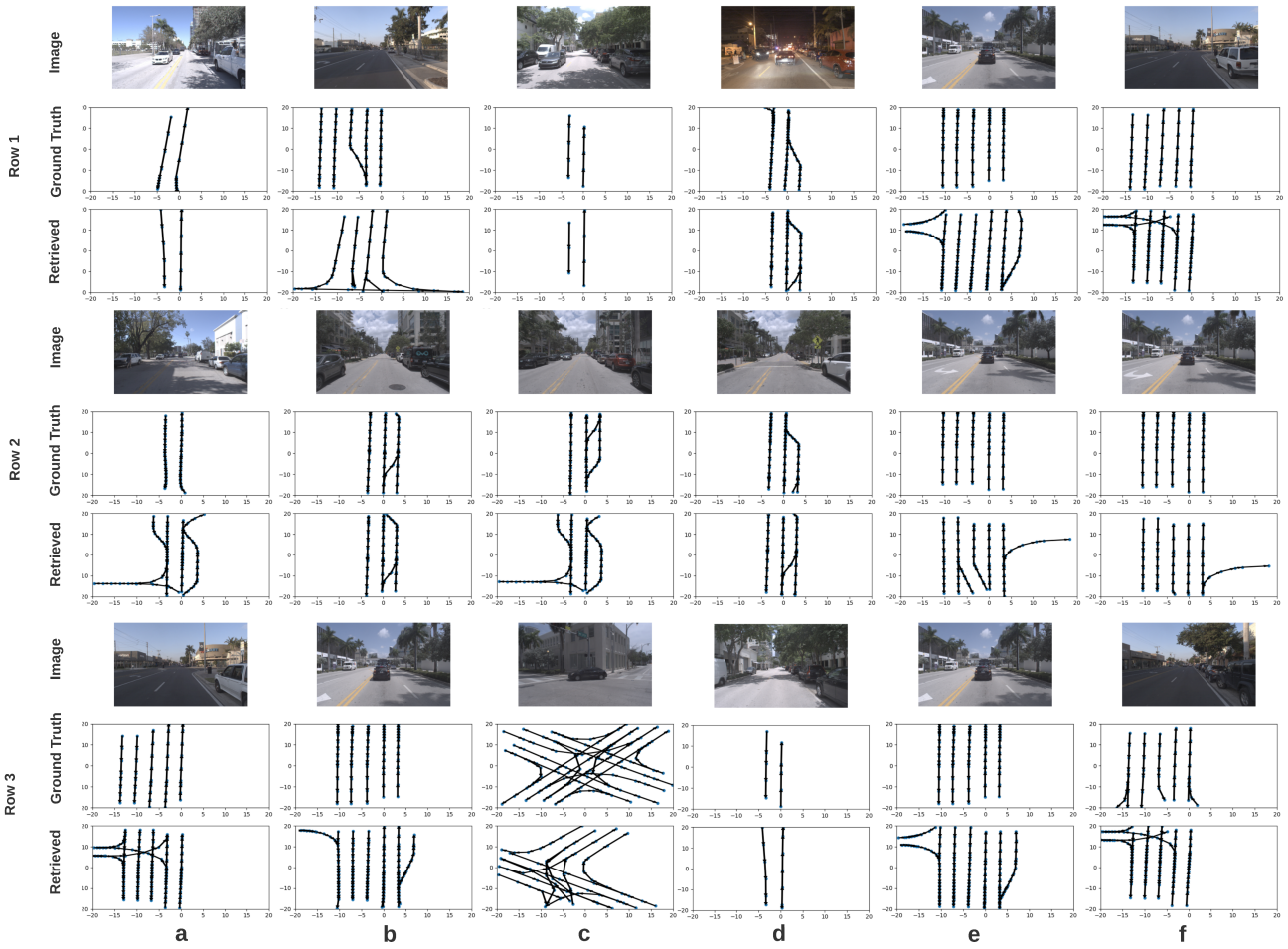

Figure 1. In this figure, we demonstrate map expansion results when training on MIA-1 and testing on MIA-2, two Argoverse-provided non-overlapping subsets of the Miami dataset. In general, we observe that the retrieved graphs are often similar to the ground truth graphs and may require small tweaks to be used.

Table 1. **Retrieval with larger map-graph libraries** Given a fixed image and graph encoder, we explore retrieval with larger map-graph libraries. Because the library requires only graphs and *not* images, we can enlarge it with all available graph topologies from Pittsburgh and Miami maps as shown in Fig.3 in the main paper (including those regions for which no images were released in Argoverse). By sampling random ego-vehicle positions in each city map, we grow the Pittsburgh-library from 5701 to 44293 and grow the Miami-library from 7421 to 112433.

#	City	Library Size 10^{-2}	Recall-metric 10^{-1}	MMD 10^{-1}	Frec. len. 10^{-2}	Frec. orien. 10^{-1}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{1}	Chamfer dist
1	PIT	5.7k	2.5075	8.9561	7.1442	1.6704	2.6949	4.3084	2.2561	1.7978
		10k	3.8970	8.4080	0.2531	0.1008	2.2379	6.0277	2.1249	0.7229
		20k	3.8234	8.1759	0.3328	0.2522	2.2296	6.4998	2.1100	0.7899
		30k	3.8221	8.1271	0.1983	0.06579	2.1928	6.4172	2.0927	0.6956
		40k	3.8031	8.2881	0.1914	0.08093	2.1352	6.0076	2.0013	0.6351

Table 2. We consider each possible combination of: 1) including a graph-adjacency-based attention mask (denoted with ✓ in the "Attn. Mask" column) as opposed to a default fully-connected transformer attention mask and 2) the inclusion of the corresponding row of the attention matrix as a node input feature for the model.

#	Att. Mask	Adj. Matrix	Recall-metric 10^{-2}	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1			2.7369	9.5924	7.1448	**1.1292**	1.9789	6.4212	3.4137	1.9574
2	✓		**2.5075**	**8.9561**	**7.1442**	1.6704	2.6949	4.3084	**2.2561**	**1.7978**
3		✓	3.6645	9.7315	10.458	33.1571	2.1423	**3.9922**	2.8453	2.2871
4	✓	✓	2.6964	9.1803	7.7164	11.0329	**1.9755**	4.6797	3.8207	2.2815

Table 3. We consider excluding the edge loss as a training objective for Pix2Map and find that it significantly worsens most metrics.

#	Edge loss	Recall-metric 10^{-2}	MMD 10^{-1}	Frec. len. 10^{-1}	Frec. orien. 10^{-2}	U. density 10^{-1}	U. reach 10^{-1}	U. conn. 10^{-1}	Chamfer dist 10^{1}
1	✗	3.0031	9.0357	9.1080	3.9042	**1.6887**	**3.1905**	2.5864	2.2093
2	✓	**2.5075**	**8.9561**	**7.1442**	**1.6704**	2.6949	4.3084	**2.2561**	**1.7978**

Figure 2. We show initial qualitative results for Map2Pix, the task of synthesizing ego-camera views given a street graph. Such synthesis may be useful for simulator-based training and validation of autonomous stacks. Because a single street geometry (of say, a two-lane road curving to the right) might retrieve multiple realistic imagery, we show the top $K = 5$ retrieved front-views.