Energy-Based Models for Cross-Modal Localization using Convolutional Transformers

Alan Wu[1] and Michael S. Ryoo[2]

***Abstract*— We present a novel framework using Energy-Based Models (EBMs) for localizing a ground vehicle mounted with a range sensor against satellite imagery in the absence of GPS. Lidar sensors have become ubiquitous on autonomous vehicles for describing its surrounding environment. Map priors are typically built using the same sensor modality for localization purposes. However, these map building endeavors using range sensors are often expensive and time-consuming. Alternatively, we leverage the use of satellite images as map priors, which are widely available, easily accessible, and provide comprehensive coverage. We propose a method using convolutional transformers that performs accurate metric-level localization in a cross-modal manner, which is challenging due to the drastic difference in appearance between the sparse range sensor readings and the rich satellite imagery. We train our model end-to-end and demonstrate our approach achieves higher accuracy than the state-of-the-art on KITTI, Pandaset, and a custom dataset.**

I. Introduction

The ability to perform accurate localization within a given environment is key to vehicle navigation. Range sensors and RGB cameras have been employed on autonomous vehicles to describe its surroundings and help determine its pose on a map prior. With the growing prevalence of lidar sensors on vehicles due to its increasing affordability and its advantage over cameras in poor lighting and adverse weather conditions, lidar-based maps are often used for localization in the absence of GPS. As the immediate environment of the vehicle is described in the form of point clouds, this representation of the vehicle's state has been used in conjunction with maps built using onboard sensors of the same modality to perform localization in both 2D and 3D spaces [1]. However, maps derived from range sensors are not available for many regions of the world, as they are often time-consuming to collect and expensive to compose, often leading to limited area coverage and constrained access. As a compelling alternative that is widely available and easily accessible, satellite imagery covers more regions of the world and contains rich structural information that can be correlated with lidar data.

[1]Alan Wu is with MIT Lincoln Laboratory, Lexington, MA, USA Alan.Wu@ll.mit.edu

[2]Michael S. Ryoo is with the Department of Computer Science, Stony Brook University, Stony Brook, NY, USA mryoo@cs.stonybrook.edu

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: 6834. This material is based upon work supported by the Department of the Army under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army.

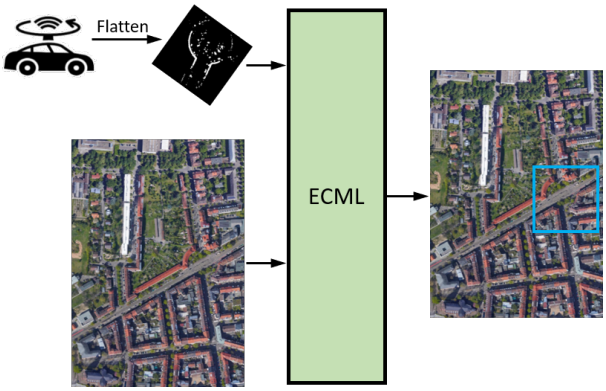


Fig. 1: Overview of our system: Energy-based Cross-Modal Localization (ECML). Vehicle is localized against satellite imagery via lidar point clouds flattened into rotated birds-eye view (BEV) images. Candidate satellite tiles are extracted from the map and paired with the BEV lidar image to find pose similarity: lidar-satellite pairs with high similarity achieve low energy. Here, the vehicle is located at the center of the blue box.

We address the problem of localization across modalities at the metric (i.e. pixel) level by training an Energy-Based Model (EBM) [2] to learn a similarity measure between the richly textural RGB satellite images and the sparse ground-based range sensor data (e.g., lidar). To better correlate with the structural information contained in the overhead imagery, we convert online lidar readings to birds-eye view (BEV) lidar images as done in [3], and later in [4] and [5]. The energy function maps a lidar-satellite image pair to a scalar energy that reflects the degree of similarity between their poses. This simple yet flexible framework suits our localization task particularly well in scaling to map priors of arbitrary sizes and regions of coverage.

As the backbone of EBMs can take on various forms, we leverage the recent advancements of convolutional neural networks (CNNs) and transformers for our task. The advent of the transformer architecture initially developed for natural language processing (NLP) applications [6] has greatly impacted the computer vision landscape upon the introduction of the visual transformer (ViT) [7] as it has outperformed traditional CNN architectures in the image classification task. Many works [8][9] have shown that purely transformer-based architectures achieved higher accuracy than their CNN counterparts, but required significantly more data to train. The authors of ViT postulated that the lack of inductive bias in transformers but inherent to CNNs have put it at a disadvantage when there is insufficient data. Some works

[10][11] have had success taking advantage of the benefits of both architectures by combining elements of each to form convolutional transformers (CT) to achieve high accuracy but with less data than ViT.

We follow the method pursued by [10]. Instead of dividing the input image into patches and tokenizing the image directly as in ViT, the image is first fed into convolutional layers before tokenization so that we can take advantage of the inductive bias property of CNNs as well as the highly effective attention mechanism of the transformer encoder. This is important to our task of correlating lidar BEV and satellite images that relies on retaining structural information of input images that would otherwise have been lost at the boundaries of patches via direct image tokenization.

The main contribution of this paper is a method to localize a lidar-mounted vehicle on a satellite map prior scaled to an arbitrary size in GPS-denied environments. Our model (Fig. 1), Energy-based Cross-Modal Localization (ECML), learns to minimize the energy between a lidar-satellite pair with high pose similarity. Our contribution also includes an approach to accelerate inference for large map priors and a custom dataset collected in desert regions of southeastern California, a sandy environment prone to dust storms and yielding conditions that are not typically present in publicly available datasets. We validate our approach on the widely used KITTI [12] dataset, the more recent Pandaset [13] dataset, and our custom dataset.

II. Related Work

A. Cross-modal Localization with Overhead Imagery

Localization using overhead imagery is a well studied subject. Various sensors such as cameras [14][15][16][17], lidars [18][19][20], and radars [5][21] have been used against pre-built map priors. A common approach shared by many works is to match features between a query image from the perspective of the vehicle and an aerial map image. However, much effort has gone into handcrafting the features for specific scenarios that do not generalize well to unseen cases. Some approaches [22][23] extracted edge features of buildings and applied chamfer matching on geometric shapes and lines. Relying heavily on road lane markings, [14] extracted SURF features from known landmarks while [15] used a labor intensive method based on Canny edge detection that required manual removal of false positives. More recently, various sensor modalities have been used to match against custom lidar intensity maps such as synthesized camera viewpoints [16], satellite images [24], and camera depth [18]. We use readily available satellite maps that require minimal processing.

B. Learning-based Localization Approaches

Numerous works have applied learning-based approaches to solving the problem of outdoor localization with prior maps. These can be broadly divided into three categories: pose estimation from visual and range sensor odometry, general place recognition, and metric level localization.

1) Pose Estimation: Among the earlier works to apply deep learning to localization, PoseNet [25] predicted camera pose from images showing scenes that had been previously seen during end-to-end training, but from a different perspective. This relocalization task was achieved from regressing pose directly from egocentric images. Several extensions have improved upon performance by modeling pose uncertainty [26][27], and introducing geometry-aware loss functions [28]. L3-Net [29] used point cloud data to learn pose from local keypoint descriptors through PointNet [30] followed by 3D CNNs. More recently, [31] extracted segments from lidar scans to learn segment-level descriptors to match geometric segments of maps stored in a database. The sharper geometry resulting from sparse segments augmented with image patch descriptors have yielded impressive performance. Sharper geometry from sparsity of segmented data is analogous to our BEV of lidar data, which makes the data more sparse, but yields better geometry. Similar to our approach, it used *one-shot* localization, as opposed to requiring a sequence of inputs when implementing a particle filter, for example. However, these approaches did not use prior maps, but rather built the maps while collecting their data, whereas we leverage widely available satellite imagery as our map.

2) General Place Recognition and Geolocalization: Localization through image geo-tagging is another popular line of research, where images are geo-tagged and then used as a reference for online images during inference. Some works [32] and [33] explored cross-view general place recognition by embedding representations of ground-level and aerial images using a Siamese network. Similarly, [34] and [35] learned dense features for ground-to-air geo-localization. Wang et. al [36] trained a triplet network to learn features from images and point-of-interest markers for various urban neighborhoods. Chu et al. [37] built a feature dictionary and learned projection matrices to compute a pose probability distribution of the input camera view with respect to a satellite map prior. Several works used a particle filter localization framework to match ground and aerial images. FLAG [38] learned features from detecting vertical structures, [39] used image descriptors from a warped BEV image, and [40][41] trained a Siamese network to learn common features of ground-satellite pairs while [42] used a visual transformer with triplet loss. Implementations of particle filters, such as these works, often assume prior knowledge of traversable areas such as roads to limit their initial search to these areas in order to achieve convergence, whereas we do not assume any prior knowledge of the map contents. In addition, these works did not localize to the metric level.

3) Metric Localization Using Overhead Imagery: There has been much interest recently in applying deep learning techniques to metric localization. Ma et al. [43] leveraged high-definition (HD) semantic maps from lidar point clouds to localize the vehicle on highways using lane markers and road signs. Barsan et al. [3] instead used unannotated lidar intensity maps and [44] investigated using both types of HD maps, but more efficiently by compressing them into

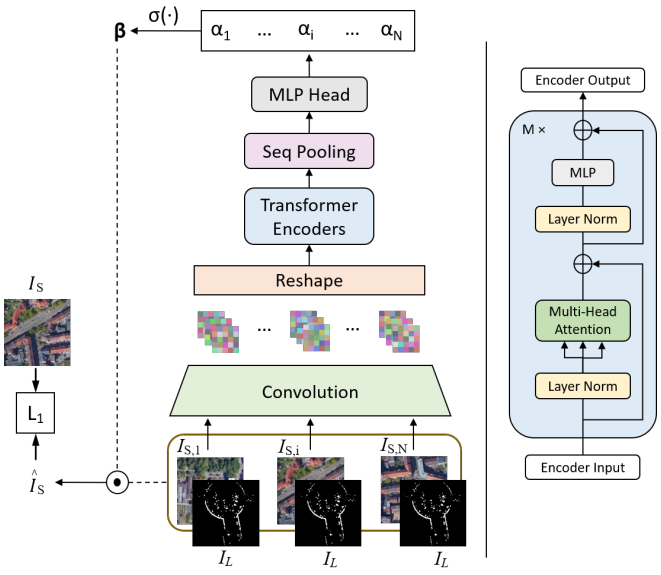


Fig. 2: Our cross-modal localization network ECML-CT. Birds-eye view (BEV) lidar images concatenated ($\oplus$) with an array of candidate satellite tiles serve as input to the model. We construct $\hat{I}_S$ from the dot product $\boldsymbol{\beta} \cdot \mathbf{I}_S$ where the coefficients $\boldsymbol{\beta}$ are the softmax output of similarity scores $\mathbf{A}$ and the satellite array $\mathbf{I}_S$ is derived from $I_{S_{map}}$. At inference, we simply select the lidar-satellite pair that yields the highest similarity score α^* without using the softmax ($\sigma(\cdot)$) or the dot product ($\odot$).

binary representations. All these works, however, localize against maps built using the same modality as their online sensor. Others [4][5] employed overhead imagery as their map prior while using either a radar or lidar onboard sensor to perform cross-modal localization. Similar to [3] and [29], they searched exhaustively for the optimal pose offset instead of direct regression. Our approach differs in two main aspects: (1) we do not require GPS at inference and (2) our map is much greater in size than the online BEV image whereas theirs is of similar size but slightly offset due to a noisy GPS. While their modalities have significant overlap, our map prior can be orders of magnitude greater than the online image, leading to a much larger search space.

C. Convolutional Transformers

The predominant architecture in NLP for sequence-to-sequence mapping, the transformer [6] inspired the visual versions that have outperformed CNN counterparts in image classification [7], object detection [45][46], anomaly detection [47], semantic segmentation [48][49], and autonomous driving [50] tasks. However, pure visual transformers require substantially more data than CNNs, partly because of their lack of inductive bias [7][10]. Various works have attempted to alleviate the need for large amounts of data by using strong data augmentation and knowledge distillation [51] or by incorporating convolutional layers to extract the lower level features of the image before tokenizing the feature maps and using the transformer to learn higher level features [48][11][10]. Our model builds upon [10] and, to our best knowledge, is the first to leverage transformers for cross-modal localization.

III. Methodology

A. Energy-Based Models

Energy-Based Models [2] provide a simple yet powerful and flexible framework that does not require normalization, but rely on a scalar energy to capture dependencies between variables. Their effectiveness has been demonstrated in multiple domains including image generation [52][53], classification [54], object detection [55], machine translation [56], structured prediction [57][58], reinforcement learning [59], and continual learning [60]. [61] has done work on object localization and anomaly detection, but not vehicle localization using range sensors. To our best knowledge, cross-modal localization is a novel application for EBMs.

Energy-Based Models rely on the principle that any probability density $p(\mathbf{u})$ for $\mathbf{u} \in \mathbb{R}^K$ can be expressed as the Boltzmann distribution:

$$p_{\mathbf{w}}(\mathbf{u}) = \frac{\exp(-E_{\mathbf{w}}(\mathbf{u}))}{Z(\mathbf{w})} \ , \ Z(\mathbf{w}) = \int_{\mathbf{u}} \exp(-E_{\mathbf{w}}(\mathbf{u})) \quad (1)$$

where $E_{\mathbf{w}}(\mathbf{u}) : \mathbb{R}^K \rightarrow \mathbb{R}$, known as the *energy function*, maps each data point to a scalar energy value, and $Z(\mathbf{w})$ is the normalizing partition function. Thus, one can parameterize an EBM using any function that takes $\mathbf{u}$ as the input and returns a scalar. In our application, the energy function $E_{\mathbf{w}}(\mathbf{u})$ is a convolutional transformer parameterized by $\mathbf{w}$.

B. Energy-Based Models for Cross-Modal Localization

We use lidar data to find vehicle pose within a satellite image. Point cloud data from range sensors have shown to be effective in localization tasks when a flattening step is applied to produce a birds-eye view (BEV) of the environment surrounding the vehicle. [3] We use this BEV representation to localize within a large map area. While our primary objective is to estimate vehicle position in the x,y-plane, our method also solves for the rotational offset between the online lidar image and the satellite map prior.

Given a greyscale lidar image $I_L \in \mathbb{R}^{d \times d \times 1}$ and an RGB satellite map $I_{S_{map}} \in \mathbb{R}^{D \times D \times 3}$ ($D >> d$), we seek to find the (x, y) pixel location of the vehicle on $I_{S_{map}}$. To accomplish this, we learn an energy function $E_{\mathbf{w}}(I_L, I_{S_i})$ where $I_{S_i} \in \mathbb{R}^{d \times d \times 3}$ is the i-th image cropped from $I_{S_{map}}$ at coordinate (x_i, y_i). Since there may be rotational offsets between I_L and $I_{S_{map}}$, a set of N_θ rotated lidar images $I_L(\theta)$ serves as candidates to describe the state of the vehicle. Therefore, there is a total of $N_S = M_S \times N_\theta$ lidar-satellite image pairs, where M_S is the number of satellite images cropped from $I_{S_{map}}$ via a sliding window. The pose of the vehicle is determined by the lowest energy achieved among all lidar-satellite pairs:

$$(x^*, y^*, \theta^*) = \underset{x,y,\theta \in \mathcal{X},\mathcal{Y},\Theta}{\arg\min} E_{\mathbf{w}}(I_L(\theta), I_S(x, y)) \quad (2)$$

We adapt the general formulation of EBMs from Eq. 1 for cross-modal localization. Given input I_L and a discrete

set $\mathcal{I}_S$ of satellite tiles $\mathbf{I}_S$ derived from satellite map $I_{S_{map}}$, we use the Boltzmann distribution to define the conditional likelihood of tile I_S given I_L as follows:

$$p_{\mathbf{w}}(I_S|I_L) = \frac{\exp(-E_{\mathbf{w}}(I_L, I_S))}{Z(\mathbf{w}; I_L)}, \quad (3)$$
$$Z(\mathbf{w}; I_L) = \sum_{I'_S \in \mathcal{I}_S} \exp(-E_{\mathbf{w}}(I_L, I'_S))$$

where $E_{\mathbf{w}}(I_L, I_S) : (\mathbb{R}^U, \mathbb{R}^V) \rightarrow \mathbb{R}$ is the energy function mapping a lidar-satellite pair (I_L, I_S) to a scalar energy value, and $Z(\mathbf{w}; I_L)$ is the partition function for normalization. The objective of the EBM is to minimize energy at ground truth tile I_S and increase the energy at other tiles I'_S.

The key component of our EBM is a convolutional transformer (CT) adopted from [10]. The CT architecture applies particularly well to our application as the early convolutional layers help better preserve local structural information that would otherwise have been lost by direct tokenization of input images (i.e. patch boundaries). The class and position embeddings used in the ViT architecture are eliminated in the CT approach. The implementation of the sequence pooling layer [10] renders such embeddings unnecessary, leading to a more compact transformer. We investigated two EBM architectures: with a CT backbone (ECML-CT) and with a CNN backbone (ECML-CNN). We detail the architecture of ECML-CT in the following section and describe ECML-CNN along with our other baselines in the Appendix.

C. Training ECML-CT

To estimate $E_{\mathbf{w}}(I_L, I_S)$, we train the convolutional transformer (CT) in a self-supervised manner to construct the predicted satellite tile $\hat{I}_S$ from a weighted ensemble of tile candidates. The model architecture for ECML-CT is shown in Fig. 2. It takes as input a lidar-satellite pair $(I_{LS_i} = I_L \oplus I_{S_i})$ concatenated along the channel axis where $I_{LS_i} \in R^{H \times W \times C}$. We use two convolutional layers with ReLU activation and max pooling to obtain an intermediate representation $z \in R^{H_0 \times W_0 \times p}$ so that p is equivalent to the embedding dimension of the transformer backbone used in ViT. We reshape z to $z_0 \in R^{l \times p}$ where $l = H_0 \times W_0$, which is the length of the embedding sequence. z_0 is then fed to the transformer encoder. The sequential output of the M-layer transformer encoder is followed by a sequence pooling step [10] where the importance weights are assigned for the output sequence, which is subsequently mapped to a similarity score for the lidar-satellite pair in the MLP head.

Each batch consists of proximal and distant lidar-satellite pairs with N_S pairs in total, and therefore yielding $\mathbf{A} = [\alpha_1, \alpha_2, ...\alpha_{N_S}]^T \in \mathbb{R}^{N_S}$, which we use as similarity scores at inference. To minimize the energy, which is the negative output (-α_i) of the CT, we aim to generate a satellite tile $\hat{I}_S$ from a weighted sum of the candidate satellite tiles $\mathbf{I}_S$ by applying a softmax operation to $\mathbf{A}$ to obtain a normalized probability distribution $\boldsymbol{\beta}$ such that $\sum_{i=1}^{N_S} \beta_i = 1$. The elements of $\boldsymbol{\beta}$ serve as coefficients for a linear combination of satellite candidates $\mathbf{I}_S$ in constructing the predicted satellite image $\hat{I}_S = \boldsymbol{\beta} \cdot \mathbf{I}_S$, with the aim that the β_i for the vehicle location will be close to 1 while others will be close to 0. The likelihood of each candidate reflects the match quality between lidar-satellite image pairs, (I_L, I_{S_i}), where I_L is the aligned online lidar image and I_{S_i} is a satellite tile candidate derived from $I_{S_{map}}$. We apply L_1 loss to the predicted satellite image: $\mathcal{L}(\hat{I}_S, I_S) = ||I_S - \hat{I}_S||_1$, where I_S is the true satellite tile centered at the vehicle location.

Fig. 17 shows samples of images generated by ECML-CNN and ECML-CT, as well as true satellite images. Clearer generated images lead to higher accuracy pose predictions. We also measure the structural similarity (SSIM) and the peak signal to noise ratio (PSNR) of the generated images across 100 waypoints of each dataset, which is summarized in Table I. Higher is better.

	SSIM		PSNR	
	CT	CNN	CT	CNN
KITTI	0.60	0.46	21.40	19.14
PandaSet	0.53	0.36	20.82	20.68
Custom A	0.79	0.58	25.84	21.96
Custom B	0.52	0.28	20.00	18.44

TABLE I: Assessment of image generation using SSIM and PSNR on our datasets. Higher scores indicate more accurate predictions.

D. Vehicle Pose Inference

Since CTs and CNNs are not equivariant to rotation, we provide N_θ rotated lidar BEV candidates $I_L(\theta)$ at inference. The lidar image with the optimal rotation $I_L(\theta^*)$ paired with the satellite image at the vehicle location $I_S(x^*, y^*)$ will provide the highest score α^* of the score vector $\mathbf{A} \in \mathbb{R}^{N_S}$ where $N_S = M_S \times N_\theta$ and M_S is the number of satellite images cropped from $I_{S_{map}}$ via a sliding window. Vehicle pose is then found using Eq. 9. Note that the vector size of $\mathbf{A}$ is not limited to N_S (i.e. the number of satellite images of $\mathbf{I}_S$) at inference as we take the $argmax$ of the scores to get optimal pose estimate, and no longer use the softmax or the dot product.

As the map grows in size, sliding a window across every possible pixel would prohibit real-time inference. We propose a 2-stage inference approach that significantly decreases processing time. In the first stage, we compose a set of input pairs by skipping m pixels in both x and y coordinates, and n degrees, yielding $\frac{M_S}{m^2} * \frac{N_\theta}{n}$ pairs. We then take the top k candidates with the highest similarity scores and perform a 2nd stage of inference where we sweep the area around those candidates in pixel and angle space in 1 pixel and 1° increments, respectively. That is, we "fill the gaps" from the 1st stage around the top k candidates by sweeping $[x_j - (m-1), x_j + (m-1)]$, $[y_j - (m-1), y_j + (m-1)]$, and $[\theta_j - (n-1), \theta_j + (n-1)]$ where $j \in [1, k]$. The predicted pose is then derived from the lidar-satellite pair with the highest similarity score of the 2nd stage. We experimented with $m \in [1, 8]$ and $n \in [1, 4]$ and found inference can be performed in 1.59 sec on a Titan X GPU for an area of 351×351 m^2. Fig. 4c shows pose errors and the corresponding inference time per sample for various m and n values.

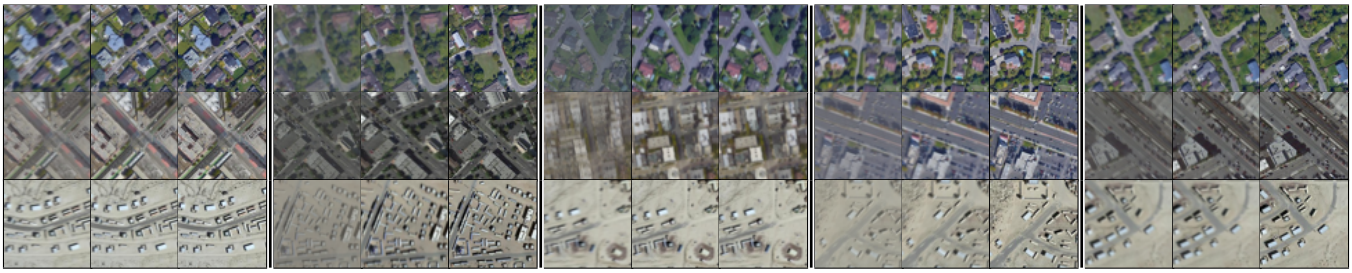
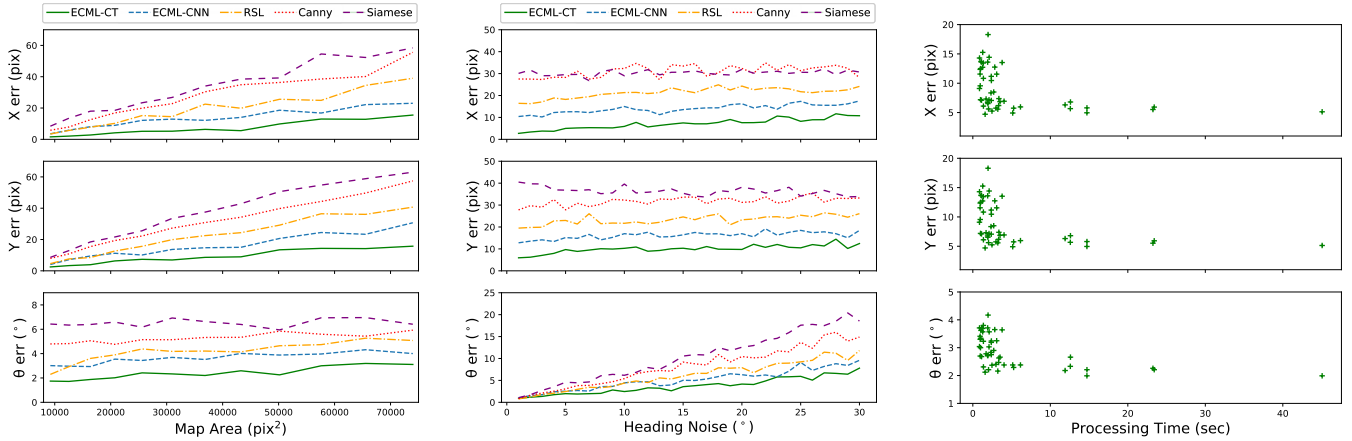


Fig. 3: Each 3-image set shows CNN-generated (left), CT-generated (middle), and ground truth (right) satellite tiles. Top row is KITTI; middle is PandaSet; and bottom is Custom. Clearer generated images lead to more accurate pose predictions.



(a) (b) (c)

Fig. 4: (a) Pose errors for various methods plotted against satellite map sizes on the KITTI dataset with $\pm 10^{\circ}$ of heading noise. (b) Pose errors plotted against online heading uncertainty. (c) Pose errors for various values of m and n plotted against inference processing time, for map size of 351×351 m^2 and heading noise of $\pm 10^{\circ}$.

IV. Experiments and Results

A. Datasets and Experimental Setup

We use the publicly available KITTI dataset [12], Pandaset dataset [13], and a custom dataset to validate our approach. While KITTI is a well-known and popular dataset collected in Karlsruhe consisting of mainly suburbs, the more recently composed Pandaset is set in San Francisco, where urban canyons are more prevalent. Our custom dataset is collected in two towns on and off roads in a desert in southeastern California. Because of the arid nature of the region, dust storms and sand kicked up by other vehicles often caused visual impairment that would greatly degrade the effectiveness of RGB cameras. Our lidar sensors, however, were able to capture the surrounding environment despite the unfavorable environment. There have been work that showed that lidar is able to sense the environment even in dust storms [62] that would otherwise have stymied visibility for RGB cameras.

For KITTI, we use 5 routes for training and 2 routes for test, with the split roughly at 16000 images and 1600 images, respectively. During inference, we use every 16th frame, which yields 100 waypoints along the route. For Pandaset, we use 93 sequences for training and 10 sequences for test; we follow a similar split ratio (7440 training images and 800 test images). For our custom dataset, we split our data in two different ways. In the first scenario (custom A), we combine data from both towns and use 205 segments for training and 47 segments for test, which amounts to a split of about 58500 images and 13500 images, respectively. In the second scenario (custom B), we use data collected in one town for training and the other town for test, which is more challenging since both the lidar data and the satellite imagery will be unseen by the model, as opposed to only the lidar data. The split for this scenario is 147 segments for training and 105 segments for test, which leads to about 42000 images and 30000 images, respectively. For all the datasets we extract 100 waypoints evenly from the test split to assess pose prediction accuracy. Odometry information used for heading and GPS coordinates used to retrieve satellite images for training were made available through onboard sensors for all datasets. No GPS data is used during inference.

We convert lidar point clouds to birds-eye view (BEV) images to correlate with satellite imagery for vehicle localization. The sparse greyscale lidar BEV images, composed of only points registered above ground (i.e. $z>0$), are 64×64 pixels with a resolution of 1.83 m/pixel, which is the same scale as all satellite imagery. Individual RGB satellite tiles

Fig. 5: Examples of online vehicle lidar image queries overlaid on the bottom-left corner of satellite map priors and resulting localizations for KITTI (top), PandaSet (middle), and Custom (bottom) datasets. Black, red, and yellow boxes are ground truth, CT-predicted, and CNN-predicted locations, respectively. The maps are retrieved from a database so that the vehicle location is randomly positioned on the map to make the task more challenging and allow for *one-shot* localization.

derived from the satellite map are of the same image size and resolution. We vary the coverage area of the satellite map at each waypoint to help the model learn better for *one-shot* localization, in that adjacent vehicle waypoints would have very different (x,y) pixel locations on the maps $I_{S_{map,i}}$ and $I_{S_{map,i+1}}$, which is illustrated in the two leftmost maps on the top row of Fig. 5. Note the difference in map appearance.

To achieve metric-level localization, the position search space would span the entire satellite map minus the borders since all candidate satellite tiles would need to be wholly encapsulated within the map prior. Although an exhaustive search in the heading search space would span $[-180°, +180°]$, we assume access to a coarse heading measurement, such as a noisy compass, is available at inference. Not only does this greatly reduce the computational complexity, but also helps minimize the possibility of encountering perceptual bias from the BEV perspective. For example, a rotation of 90° at a cross-road intersection or 180° on a straight road would lead to heading ambiguity. Therefore, for our experiments, we add noise sampled uniformly from the range $[-10°, +10°]$ to the ground truth heading θ_{gt} and propose candidates rotated in $1°$ increments to cover the range of uncertainty. This results in the heading search space $\theta \in [(\theta_{gt} - 10°), (\theta_{gt} + 10°)]$.

B. Experimental Results

We compare our approach to other methods for metric localization as shown in Table III. The state of the art in cross-modal localization via live range sensing against satellite imagery that inspired our work is [4]. However, different from our goal, theirs is to refine a coarse estimate from a noisy GPS fix, which implies that their search area of the satellite map is only slightly greater than the immediate online range sensor coverage area. That is, the size of their satellite map image is similar in size as the range sensor BEV image, whereas in our case the size of the satellite map image is much greater than the sensor BEV image. Having the same image sizes between the two modalities allows them to generate a synthetic image of the entire satellite map that takes on the appearance of a sensor BEV image. With both

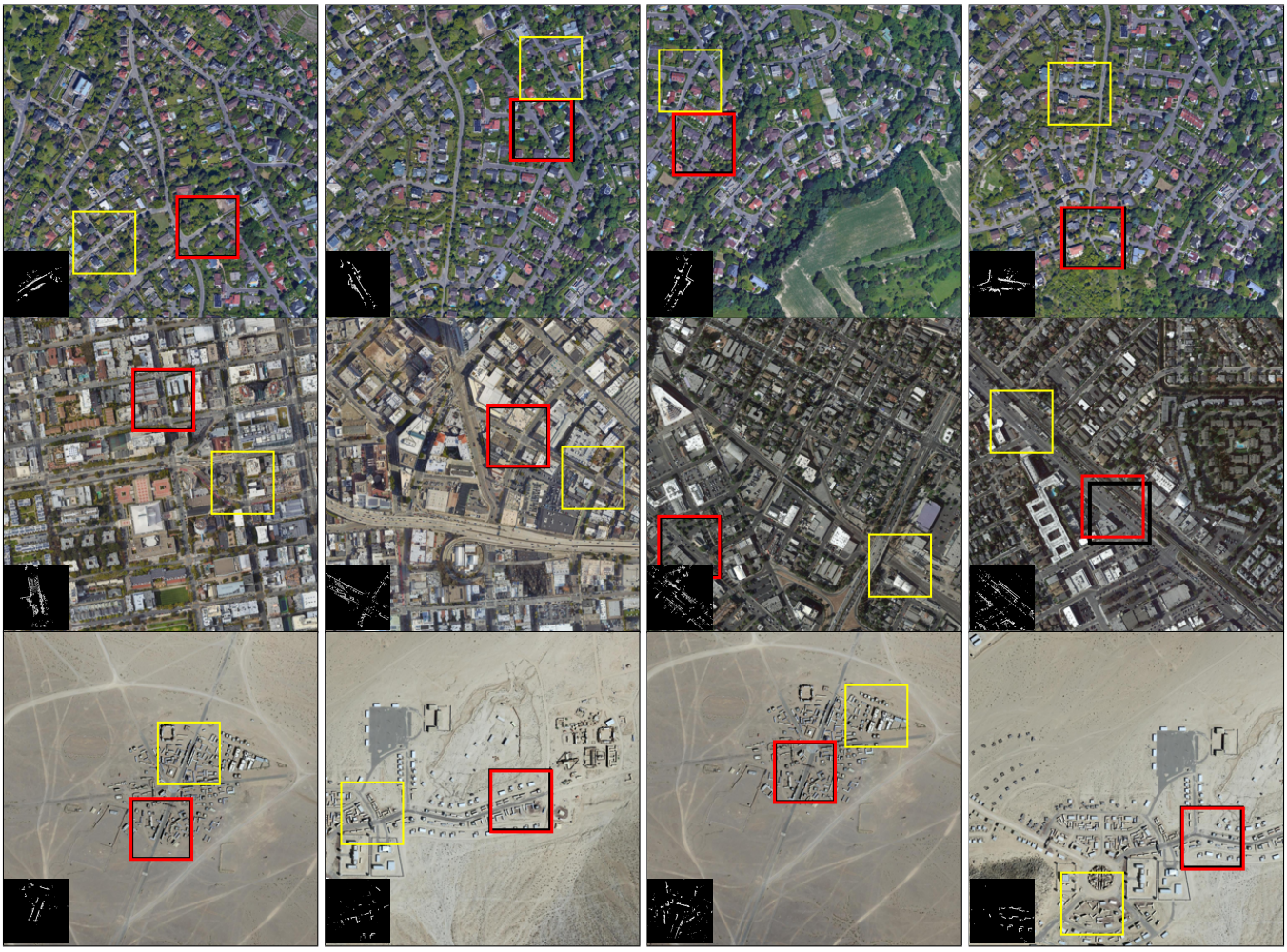


Fig. 6: Examples of ECML-CT successful predictions but ECML-CNN failures. Note the similarity in appearance between the true locations and the incorrect predictions. Black, red, and yellow boxes are true, CT-predicted, and CNN-predicted locations, respectively.

images now appearing in the same modality, the correlation surface is found by convolving the online BEV image with the synthetic BEV image, and the (x,y) position offset is found by taking the $argmax$ of the correlation matrix.

Our task is to localize the vehicle in GPS denied regions, and therefore the search space (i.e. satellite map size) is much greater than the coverage area of the online range sensor. With a satellite map size much greater than the range sensor BEV image size, we can no longer generate a range sensor-like synthetic image of the entire satellite map as a whole as done in [4] and [5]. To adapt their approach to our task, we assemble all possible satellite tiles $\mathbf{I}_S$ derived from the satellite map $I_{S_{map}}$ via a sliding window of size $d \times d$ and convert each tile to synthetic lidar images $\mathbf{I}_{L_{gen}}$ using the image generator $g(\cdot; \mathbf{w_g})$ similar to that in [4] with satellite tiles from our datasets as input: $g(\mathbf{I}_S; \mathbf{w_g}) \rightarrow \mathbf{I}_{L_{gen}}$ where $\mathbf{w_g}$ are learned parameters. Each generated lidar tile is then correlated with the rotated online lidar images $\mathbf{I}_L(\theta)$ via convolution:

$$(x^*, y^*, \theta^*) = \underset{x,y,\theta \in \mathcal{X},\mathcal{Y},\Theta}{\arg\max} \boldsymbol{I}_L(\theta) \circledast \boldsymbol{I}_{L_{gen}}(x, y) \quad (4)$$

The convolution operation ($\circledast$) can be performed more efficiently in the Fourier domain [3]:

$$I_L \circledast I_{L_{gen}} = F^{-1}\{F(I_L) \otimes F(I_{L_{gen}})\} \quad (5)$$

where $\otimes$ denotes element-wise multiplication, and F and F^{-1} indicate the Fourier and inverse Fourier transforms, respectively.

In addition to [4], we include ECML-CNN as a baseline as well as methods using Canny edges [63], metric learning using Siamese networks [41], and a naive RESNET-based classifier [64]. Instead of using street-level camera images [41], we used BEV lidar images as a Siamese network input to correlate with satellite images. We show examples of query lidar image and satellite map localization pairs in Fig. 5. Table II summarizes the localization results of our experiment showing our EBM achieving superior accuracy compared to other approaches. Specifically, ECML-CT is able to produce the most accurate pose predictions due to the self-attention mechanism within the transformer encoder to better correlate features of lidar-satellite pairs. Custom A refers to our custom dataset split after combining data from each town whereas custom B consists of training on one

	e_x (pix)	e_y (pix)	e_θ (°)
KITTI			
ECML-CT (ours)	***4.7***	***8.2***	***2.12***
ECML-CNN (ours)	12.2	14.8	3.53
RSL [4]	22.2	23.9	4.18
Canny	30.4	29.8	5.21
Siamese [41]	31.2	34.4	6.56
Classifier	36.9	38.8	6.77
Random	43.0	43.0	7.00
PandaSet			
ECML-CT (ours)	***8.8***	***9.3***	***1.70***
ECML-CNN (ours)	9.4	10.6	2.44
RSL [4]	30.0	32.9	3.27
Canny	27.2	23.2	4.80
Siamese [41]	32.6	30.0	5.63
Classifier	40.7	42.3	6.64
Random	43.0	43.0	7.00
Custom A			
ECML-CT (ours)	***3.1***	***1.6***	***1.69***
ECML-CNN (ours)	8.0	6.4	2.58
RSL [4]	29.5	29.8	1.77
Canny	19.4	16.4	4.82
Siamese [41]	31.0	27.1	5.76
Classifier	39.2	44.0	6.06
Random	43.0	43.0	7.00
Custom B			
ECML-CT (ours)	***9.7***	***5.9***	***2.46***
ECML-CNN (ours)	28.4	24.3	5.85
RSL [4]	39.8	30.8	4.96
Canny	15.3	12.3	3.89
Siamese [41]	39.9	22.4	6.18
Classifier	36.1	45.5	5.98
Random	43.0	43.0	7.00

TABLE II: Comparison of methods for mean distance error for x and y, and for mean angle error for θ on KITTI, Pandaset, Custom A, Custom B. Scale factor is 1.83 m/pixel.

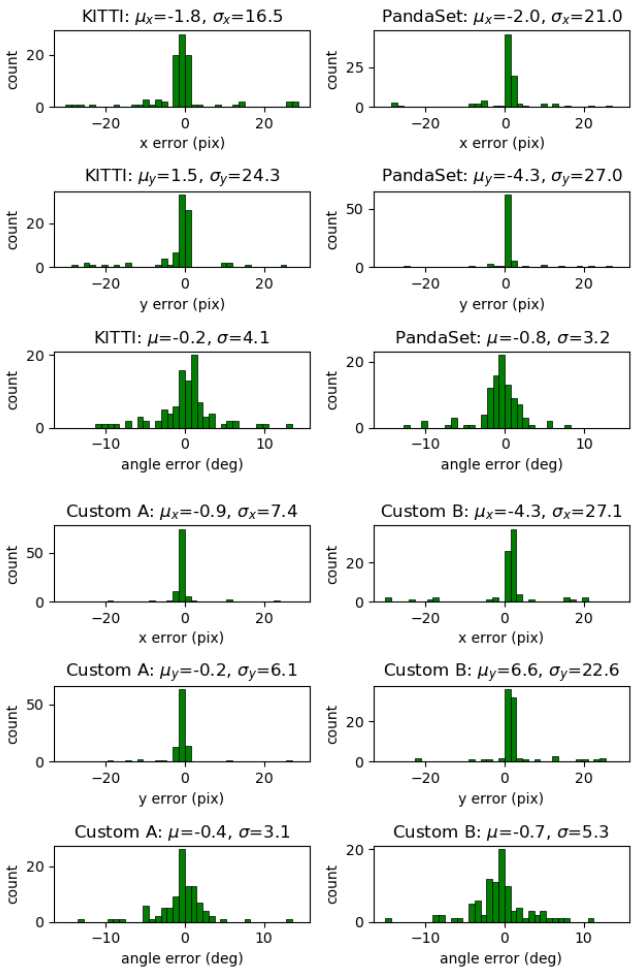


Fig. 7: Distribution of 100 samples for ECML-CT on KITTI, PandaSet, Custom A, and Custom B in x, y, and θ, along with their mean and standard deviation.

town and testing on the other, with custom B being more challenging since both lidar and satellite data are unseen, which is reflected in the table. It is also worth noting that inference error from Custom A is much lower than KITTI and Pandaset for ECML-CT, which may be explained by the lack of greenery such as trees and shrubs amongst buildings that would have made lidar-satellite correlation more difficult. Unsurprisingly, Canny edges performed quite well with the custom dataset whose buildings and roads lead to sharp lines unobstructed by greenery. In Fig. 7, histograms are shown for sample error mean and standard deviation for each dataset using ECML-CT; histograms for ECML-CNN are shown in the Appendix. One can see that ECML-CT predictions have lower pose error and are more robust (i.e. lower std dev) than ECML-CNN. The satellite maps are 351×351 m^2 (192×192 pix^2) for the results reflected in the table and the histograms. As the map area expands, the likelihood of similar structures occurring also increases, which in turns leads to higher error rates. Nevertheless, the rate of error increase for ECML is lower than other models we evaluated. This can be seen in Fig. 4a. Likewise, increasing heading noise leads to a greater search space that will include similar structures that are offset in rotation, which leads to higher pose error as shown in Fig. 4b.

V. Conclusion

We present a new method using an Energy-based Model for cross-modal localization between ground range sensors and overhead imagery in GPS-denied environments. Specifically, we demonstrate our approach leveraging convolutional transformers (ECML-CT) localizing lidar BEV images against satellite maps. Although satellite imagery is much easier to obtain than laboriously collected custom lidar maps, the drastic difference in appearance between lidar and satellite modalities makes this a very challenging task. Our model learns to assess the quality of lidar-satellite image pairs through an efficient training mechanism in order to provide accurate pose estimates for map sizes much greater than those in previous works for cross-modal localization. Even with larger map priors, we are able to perform real-time inference with a two stage approach, and demonstrate the superior performance of our model in these diverse settings.

REFERENCES

[1] C. Chen, B. Wang, C. Xiaoxuan Lu, N. Trigoni, and A. Markham, "A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence," *arXiv e-prints*, p. arXiv:2006.12567, June 2020.

[2] Y. LeCun, S. Chopra, R. Hadsell, F. J. Huang, and et al., "A tutorial on energy-based learning," in *Predicting Structured Data*. MIT Press, 2006.

[3] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun, "Learning to localize using a lidar intensity map," in *The 2nd Conference on Robot Learning (CoRL)*, 2018.

[4] T. Y. Tang, D. De Martini, D. Barnes, and P. Newman, "Rsl-net: Localising in satellite images from a radar on the ground," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1087–1094, 2020.

[5] T. Tang, D. De Martini, S. Wu, and P. Newman, "Self-supervised localisation between range sensors and overhead imagery," 07 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," vol. abs/2010.11929, 2020.

[8] S. H. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," 2021.

[9] Y. Liu, Y. Zhang, Y. Wang, F. Hou, J. Yuan, J. Tian, Y. Zhang, Z. Shi, J. Fan, and Z. He, "A survey of visual transformers," 2021.

[10] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the big data paradigm with compact transformers," *CoRR*, vol. abs/2104.05704, 2021.

[11] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu, "Incorporating convolution designs into visual transformers," 2021.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[13] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, Y. Wang, and D. Yang, "Pandaset: Advanced sensor suite dataset for autonomous driving," 2021.

[14] M. Noda, T. Takahashi, D. Deguchi, I. Ide, H. Murase, Y. Kojima, and T. Naito, "Vehicle ego-localization by matching in-vehicle camera images to an aerial image," 11 2010, pp. 163–173.

[15] O. Pink, "Visual map matching and localization using a global feature map," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, pp. 1–7.

[16] R. Wolcott and R. Eustice, "Visual localization within lidar maps for automated urban driving," *IEEE International Conference on Intelligent Robots and Systems*, pp. 176–183, 10 2014.

[17] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *International Journal of Robotics Research*, vol. 32, pp. 1645–1661, 12 2013.

[18] Y. Xu, V. John, S. Mita, H. Tehrani, K. Ishimaru, and S. Nishino, "3d point cloud map based vehicle localization using stereo camera," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 487–492.

[19] P. Carle and T. Barfoot, "Global rover localization by matching lidar and orbital 3d maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[20] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based slam using aerial images as prior information," *Autonomous Robots*, vol. 30, pp. 25–39, 01 2009.

[21] D. Barnes and I. Posner, "Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[22] K. Y. K. Leung, C. M. Clark, and J. P. Huissoon, "Localization in urban environments by matching ground level video images with an aerial image," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 551–556.

[23] A. Li, V. I. Morariu, and L. S. Davis, "Planar structure matching under projective uncertainty for geolocation," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 265–280.

[24] L. De Paula Veronese, E. de Aguiar, R. C. Nascimento, J. Guivant, F. A. Auat Cheein, A. F. De Souza, and T. Oliveira-Santos, "Re-emission and satellite aerial maps applied to vehicle localization on urban environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4285–4290.

[25] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946.

[26] A. Kendall and R. Cipolla, "Modelling Uncertainty in Deep Learning for Camera Relocalization," *arXiv e-prints*, p. arXiv:1509.05909, Sept. 2015.

[27] M. Cai, C. Shen, and I. Reid, "A hybrid probabilistic model for camera relocalization," in *The British Machine Vision Conference (BMVC)*, 2018.

[28] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6555–6564.

[29] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based lidar localization for autonomous driving," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6382–6391.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *arXiv e-prints*, p. arXiv:1612.00593, Dec. 2016.

[31] S. Ratz, M. Dymczyk, R. Siegwart, and R. Dubé, "OneShot Global Localization: Instant LiDAR-Visual Pose Estimation," *arXiv e-prints*, p. arXiv:2003.13279, Mar. 2020.

[32] S. Workman, R. Souvenir, and N. Jacobs, "Wide-Area Image Geolocalization with Aerial Reference Imagery," *arXiv e-prints*, p. arXiv:1510.03743, Oct. 2015.

[33] T. Lin, Yin Cui, S. Belongie, and J. Hays, "Learning deep representations for ground-to-aerial geolocalization," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5007–5015.

[34] S. Hu, M. Feng, R. M. H. Nguyen, and G. H. Lee, "Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7258–7267.

[35] S. Hu and G. H. Lee, "Image-Based Geo-Localization Using Satellite Imagery," *arXiv e-prints*, p. arXiv:1903.00159, Mar. 2019.

[36] Z. Wang, H. Li, and R. Rajagopal, "Urban2Vec: Incorporating Street View Imagery and POIs for Multi-Modal Urban Neighborhood Embedding," *arXiv e-prints*, p. arXiv:2001.11101, Jan. 2020.

[37] H. Chu, H. Mei, M. Bansal, and M. R. Walter, "Accurate Vision-based Vehicle Localization using Satellite Imagery," *arXiv e-prints*, p. arXiv:1510.09171, Oct. 2015.

[38] X. Wang, S. Vozar, and E. Olson, "Flag: Feature-based localization between air and ground," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3178–3184.

[39] A. Viswanathan, B. Pires, and D. Huber, "Vision based robot localization by ground to satellite matching in gps-denied situations," *IEEE International Conference on Intelligent Robots and Systems*, pp. 192–198, 10 2014.

[40] D. Kim and M. R. Walter, "Satellite image-based localization via learned embeddings," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2073–2080.

[41] L. M. Downes, D.-K. Kim, T. J. Steiner, and J. P. How, "City-wide street-to-satellite image geolocalization of a mobile ground agent," 2022.

[42] S. Zhu, M. Shah, and C. Chen, "Transgeo: Transformer is all you need for cross-view image geo-localization," 2022.

[43] W. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun, "Exploiting sparse semantic hd maps for self-driving vehicle localization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[44] X. Wei, I. A. Bârsan, S. Wang, J. Martinez, and R. Urtasun, "Learning to localize through compressed binary maps," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[46] Y. Wang, V. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "DETR3D: 3d object detection from multi-view images via 3d-to-2d queries," 2021.

[47] P. Mishra, R. Verk, D. Fornasier, C. Piciarelli, and G. L. Foresti, "Vt-adl: A vision transformer network for image anomaly detection

and localization," in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, 2021.
[48] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, M. Tomizuka, K. Keutzer, and P. Vajda, "Visual transformers: Token-based image representation and processing for computer vision," 2020.
[49] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
[50] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
[51] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," 2020.
[52] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, "A theory of generative convnet," 2016.
[53] E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu, "On the anatomy of mcmc-based maximum likelihood learning of energy-based models," 2019.
[54] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky, "Your classifier is secretly an energy based model and you should treat it like one," 2019.
[55] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön, "How to train your energy-based model for regression," 2020.
[56] L. Tu, R. Y. Pang, S. Wiseman, and K. Gimpel, "Engine: Energy-based inference networks for non-autoregressive machine translation," 2020.
[57] A. Rooshenas, D. Zhang, G. Sharma, and A. McCallum, "Search-guided, lightly-supervised training of structured prediction energy networks," 2018.
[58] M. Gygli, M. Norouzi, and A. Angelova, "Deep value networks learn to evaluate and iteratively refine structured outputs," 2017.
[59] Y. Du, T. Lin, and I. Mordatch, "Model based planning with energy based models," 2019.
[60] S. Li, Y. Du, G. M. van de Ven, A. Torralba, and I. Mordatch, "Energy-based models for continual learning," 2020.
[61] E. U. Genc, N. Ahuja, I. J. Ndiour, and O. Tickoo, "Energy-based anomaly detection and localization," 2021.
[62]
[63] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv e-prints*, p. arXiv:1512.03385, Dec. 2015.
[65] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.
[66] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *arXiv e-prints*, p. arXiv:1611.07004, Nov. 2016.

VI. APPENDIX

A. Map Scaling and Data Sampling

In order for our model to learn a similarity measure between lidar-satellite pairs, it is imperative that the training samples include a mixture of physically proximal and distant samples. In this regard a naive approach would be to treat only the satellite tile at the exact location of the lidar reading as the proximal sample and tiles that have no overlap with the lidar image as distant samples. This way prevented the model from learning as the difference was too extreme. We found that the network needed to see a gradual change in physical overlap between the modalities (i.e. similarity change) in order to learn the similarity measure as it is just as important to learn when the candidate tile is "near" as when it is at the actual location. Therefore, this may be made possible by sampling the entire satellite map with a window that slides across every pixel. However, it is important that our method scales with map size, so as the size of $I_{S_{map}}$ grows, we are challenged by heavy memory usage at training if aerial images $\mathbf{I}_S$ are densely assembled. Memory consumption increases at the rate of O(n^2) with the size of the map, leading to a dramatic increase in batch size and training time. For our application, the batch size is dictated by the size of $\mathbf{I}_S$ in generating $\hat{\mathbf{I}}_S = \boldsymbol{\beta} \cdot \mathbf{I}_S$ and must fit in the available GPU memory. While it is possible to decrease the size of $\mathbf{I}_S$ by skipping every m pixels while sliding the window, this approach led to significantly worse performance even if $m = 2$. We discovered that a combination of dense sampling of proximal regions and coarse sampling of distant regions led to optimal learning.

To compose proximal samples, we select a patch on the map centered on the true location of size $\kappa d \times \kappa d$ where $\kappa > 1$. On this patch, we slide the window of size $d \times d$ across every pixel in both the horizontal and vertical directions to assemble $\mathbf{I}_{Sfine}$. To compose distant samples, we sample N random tiles of size $d \times d$ from the map outside of the proximal patch to assemble $\mathbf{I}_{Scoarse}$. Together, $\mathbf{I}_{Sfine}$ and $\mathbf{I}_{Scoarse}$ are combined and randomly scrambled to form $\mathbf{I}_S$ = [$\mathbf{I}_{Sfine}$ + $\mathbf{I}_{Scoarse}$]. This way, the network is exposed to a diverse set of samples: both well aligned lidar-satellite pairs as well as misaligned or mismatched pairs, and learns to attribute high scores to the former and low scores to the latter without needing to exhaustively sample the entire map for training. For our experiments, we set $\kappa = 1.5$ and $N = 100$.

B. ECML Inference Tradeoffs

As the map grows in size, the search space becomes too large to perform real-time inference. Our 2-stage approach as described in section III-D significantly decreases processing time. To summarize, we skip pixels and degrees in the 1st stage of our search, and then we perform a fine sweep around the candidate lidar-satellite pairs with the top 10 highest scores. As can be seen in sample heatmaps in Fig. 9, high scoring pixel locations (darker pixels) resulting from compatible lidar-satellite pairings have neighbors with high scores as well, reflecting small lidar-satellite offsets also result in relatively high scores. As such, optimal locations missed in the 1st stage are often found in the 2nd stage since the high scoring neighbors would be discovered in the 1st stage. We experimented with skipping from 1 to 8 pixels in both x and y directions and from 1 to 4 degrees for heading. Skipping too many pixels (5+) or too many degrees (3+) not only led to missed optimal poses but the coarse spacing also left large areas around the top candidates to "fill" in the 2nd stage, which can cancel the time savings gained in the 1st stage. Ideally, the 2nd stage processing time would be much less than that of the 1st stage. For a map size of 192×192 pix^2, we found skipping every 3 or 4 pixels and every 1 or 2 degrees to result in an optimal tradeoff between pose accuracy and processing time (i.e. data points in the bottom-left of the plots in Fig. 8).

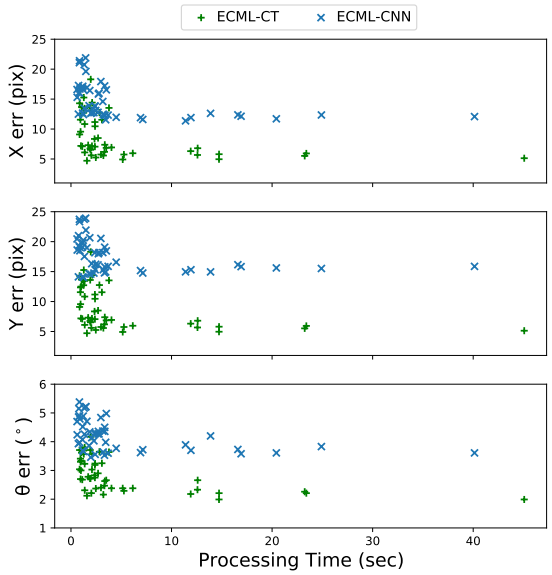


Fig. 8: Pose errors for various combinations of $m \in [1, 8]$ and $n \in [1, 4]$ plotted against inference processing time.

C. Accuracy vs Map Size

For our baseline task, we assumed access to a compass with heading noise of $\pm 10°$. In Section IV-B, we saw the effect of heading uncertainty and map size on pose error. To gain additional insight into the effectiveness of our approach, we also looked at the specific case of a low noise compass ($< 0.5°$) to see how EMCL-CT and ECML-CNN perform when we vary the map size. For simplicity, we assume there is no heading noise. As expected, position error increases as the size of the map grows, as shown in Fig. 10, where both heading noise of $\pm 10°$ (solid lines) and $0°$ (dotted lines) are plotted for competing models. We see that pose prediction error from ECML-CT grows the slowest among competing models. We note that lower heading noise for ECML-CT, ECML-CNN and RSL led to lower error as these models learned similarity measures to better distinguish heading offsets than the more naive methods using Canny edges and Siamese networks.

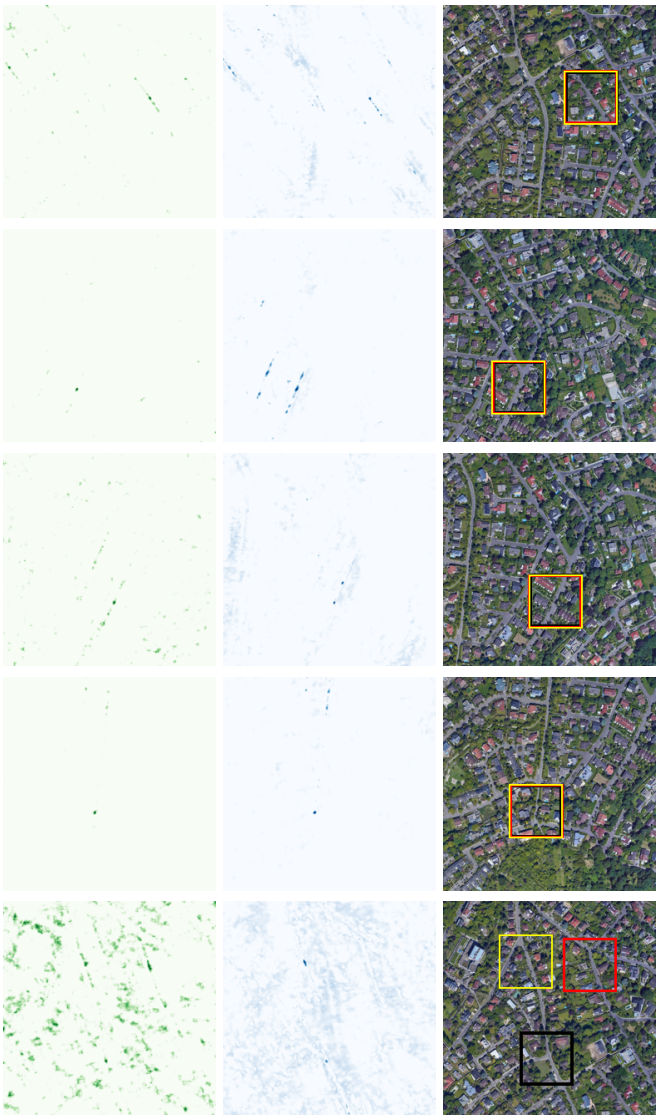


Fig. 9: Score heatmaps from ECML-CT (left/green), ECML-CNN (middle/blue) and the corresponding satellite map with vehicle localized in the center of the black (true), red (ECML-CT), yellow (ECML-CNN) boxes.

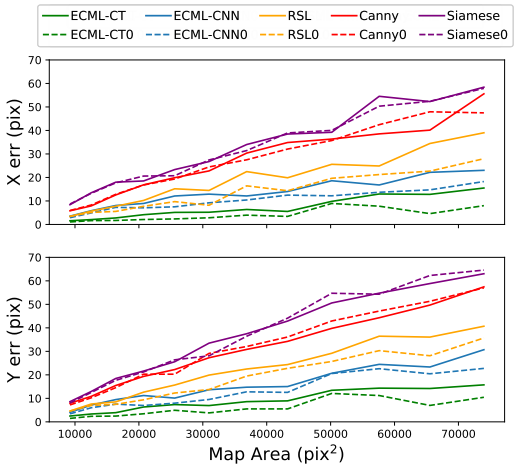


Fig. 10: Mean distance error over 100 samples for various satellite map sizes on the KITTI dataset with (solid lines) and without (dotted lines) heading noise.

D. Baseline Models

1) ECML-CNN: The CNN, shown in Fig. 12, consists of three down-convolutional layers followed by six ResNet blocks [64], where the mean is taken along the height, width, and channel dimensions to attain the network output $\mathbf{A} = [a_1, a_2, ...a_{N_S}]^T \in \mathbb{R}^{N_S}$, which we use as similarity scores at inference. To minimize the energy (-a_i), we aim to generate a satellite tile $\hat{I}_S$ from a weighted sum of the candidate satellite tiles $\mathbf{I}_S$ by applying a softmax operation to $\mathbf{A}$ to obtain a normalized probability distribution $\mathbf{B}$ such that $\sum_{i=1}^{N_S} b_i = 1$. The elements b_i serve as coefficients for a linear combination of satellite candidates $\mathbf{I}_S$ in constructing the predicted satellite image $\hat{I}_S = \mathbf{B} \cdot \mathbf{I}_S$, with the aim that the b_i for the vehicle location will be close to 1 while others will be close to 0.

2) RSL-Net: The state of the art in cross-modal localization via live range sensing against satellite imagery that inspired our work is [4]. However, different from our goal, theirs is to refine a coarse estimate from a noisy GPS fix, which implies that their search area of the satellite map is only slightly greater than the immediate online range sensor coverage area. That is, the size of their satellite map image is similar in size as the range sensor BEV image, whereas in our case the size of the satellite map image is much greater than the sensor BEV image. Having the same image sizes between the two modalities allows them to generate a synthetic image of the entire satellite map that takes on the appearance of a range sensor BEV image. With both images now appearing in the same modality, the correlation surface is found by convolving the online BEV image with the synthetic BEV image, and the (x,y) position offset is found by taking the $argmax$ of the correlation matrix.

Our task is to localize the vehicle in GPS denied regions, and therefore the search space (i.e. satellite map size) is much greater than the coverage area of the online range sensor. With a satellite map size much greater than the range sensor BEV image size, we can no longer generate a range sensor-like synthetic image of the entire satellite map as a whole as done in [4] and [5]. To adapt their approach to our task, we assemble all possible satellite tiles $\mathbf{I}_S$ derived from the satellite map $I_{S_{map}}$ via a sliding window of size $d \times d$ and convert each tile to synthetic lidar images $\mathbf{I}_{L_{gen}}$ using the image generator $g(\cdot; \mathbf{w_g})$ similar to that in [4] with satellite tiles from our datasets as input: $g(\mathbf{I}_S; \mathbf{w_g}) \rightarrow \mathbf{I}_{L_{gen}}$ where $\mathbf{w_g}$ are learned parameters. Each generated lidar tile is then correlated with the rotated online lidar images $\mathbf{I}_L(\theta)$ via convolution:

$$(x^*, y^*, \theta^*) = \underset{x,y,\theta \in \mathcal{X},\mathcal{Y},\Theta}{\arg\max} \; \boldsymbol{I}_L(\theta) \circledast \boldsymbol{I}_{L_{gen}}(x, y) \qquad (6)$$

The lidar image generator, shown in Fig. 13, follows a standard autoencoder structure similar to that used by [4].

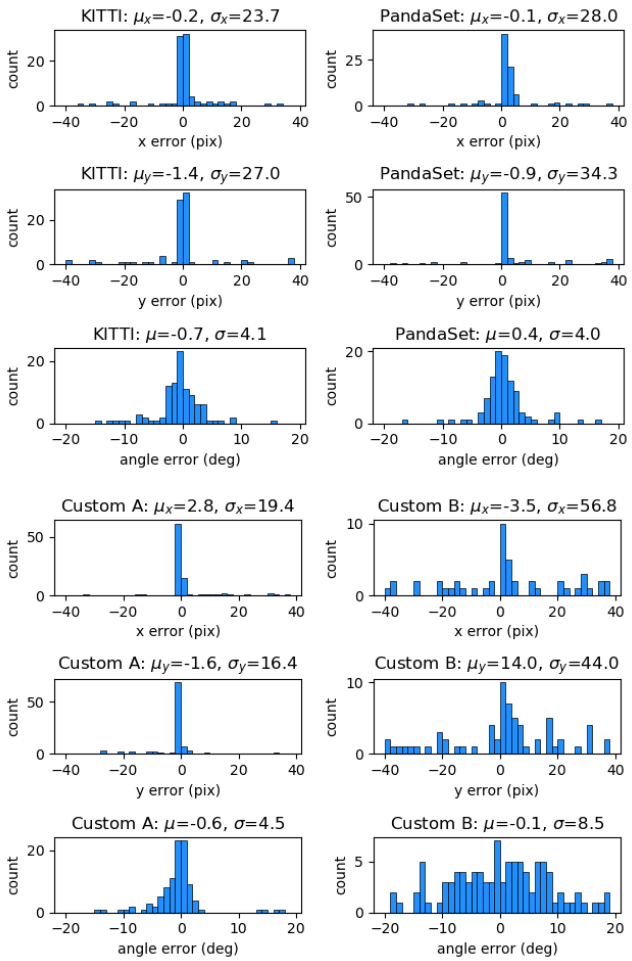


Fig. 11: Distribution of 100 samples for ECML-CNN on KITTI, PandaSet, Custom A, and Custom B in x, y, and θ, along with their mean and standard deviation.

We use slightly different network parameters since our image size is 64×64 instead of 256×256 and we only use satellite imagery for input rather than a lidar-satellite stack. As our satellite map is much larger, we encounter many regions without a lidar counterpart (ex. no lidar reading in a park or a mall). Our network has 5 down-convolutional layers with max pooling and 5 up-convolutional layers with upsampling. Each convolutional layer is followed by batch norm and leaky ReLU with a 0.2 slope coefficient.

3) Siamese Network: Metric learning has been used for cross-view localization [40][41] where ground-level camera view RGB images are used to correlate with satellite images to estimate pose. Instead of RGB camera images, we use BEV lidar images as the online input to the Siamese network to learn our embeddings Embed(I_L) and Embed(I_S). Fig. 14 shows the Siamese network where weights are shared and the output embedding of each modality is fed to a contrastive loss term:

$$L_c = \frac{1-Y}{2} D_w^2 + \frac{Y}{2} max(0, margin - D_w)^2 \quad (7)$$

where Y is the binary label and D_w is the Euclidean distance between Embed(I_L) and Embed(I_S). For our experiments, positive samples consisted of aligned lidar-satellite tile pairs that are physically near (i.e. overlap $> 25\%$). Negative samples consisted of pairs that are physically near but out of alignment (i.e. $> 3^\circ$) or physically far apart (i.e. no overlap) for both aligned and unaligned cases. We set $margin = 1$. The network consists of 3 down-convolution layers followed by 6 RESNET-blocks and 2 fully connected layers with a feature vector size of 128. Each down-convolution layer is batch-normalized and uses ReLU activation.

4) Classifier: The traditional approach to classification takes an input and predicts the class to which it belongs. The classes are pre-determined and the number of classes is fixed. Extending the classification approach to the task of localizing on a single map prior, the input would be the online lidar reading and a class would simply be defined as a location on the satellite map, which is divided into N regions (i.e classes). Specifically, each class would be an area on the map the size of a BEV lidar tile. However, since our task not only scales to any map size but also any map prior, a class must then refer to an area of a given map and not just a pre-defined map. That is, the predicted class is conditional on the map prior in addition to the online lidar reading. Our approach is illustrated in Fig. 15.

Although our implementation is non-traditional for a classifier, the classification approach is naive compared to other approaches in this paper, especially the ECML variants. For the classifier, we use a RESNET-based [64] backbone where the lidar branch has 1 down-convolution layer and the satellite map branch has 2 down-convolution layers before the output of both are concatenated along the channel axis. This is then followed by 2 down-convolution layers and then 6 RESNET-blocks and 3 fully connected layers. Each down-convolution layer is batch-normalized and uses ReLU activation. We use cross-entropy as our loss function.

E. Alternative EMCL Architectures

In our experiments with KITTI, we found that an alternative architecture leveraging the use of a generative adversarial network (GAN) [65] was able to yield marginal improvements as shown in Table III. We refer to these models as ECML-CT-G and ECML-CNN-G.

Given a lidar image $I_L \in \mathbb{R}^{d \times d \times 1}$ and a satellite map $I_{S_{map}} \in \mathbb{R}^{D \times D \times 3}$ (D >> d), we seek to find the (x,y) pixel location of the vehicle on $I_{S_{map}}$. To accomplish this, we learn $f(\cdot\,; \mathbf{w_f})$ where $\mathbf{w_f}$ are learnable parameters, such that:

$$f(I_L, I_{S_i}; \mathbf{w_f}) \rightarrow a_i \quad (8)$$

where $I_{S_i} \in \mathbb{R}^{d \times d \times 3}$ is the i-th image cropped from $I_{S_{map}}$ at coordinate (x_i, y_i), and a_i is the corresponding score for the lidar-satellite image pair. Since there may be rotational offset between I_L and $I_{S_{map}}$, a set of N_θ rotated lidar images $\mathbf{I}_{L_\theta}$ serves as candidates to describe the state of the vehicle. Therefore, we must evaluate all $M_S \times N_\theta$ lidar-satellite image pairs, where M_S is the number of satellite images cropped from $I_{S_{map}}$ via a sliding window. The location of the vehicle

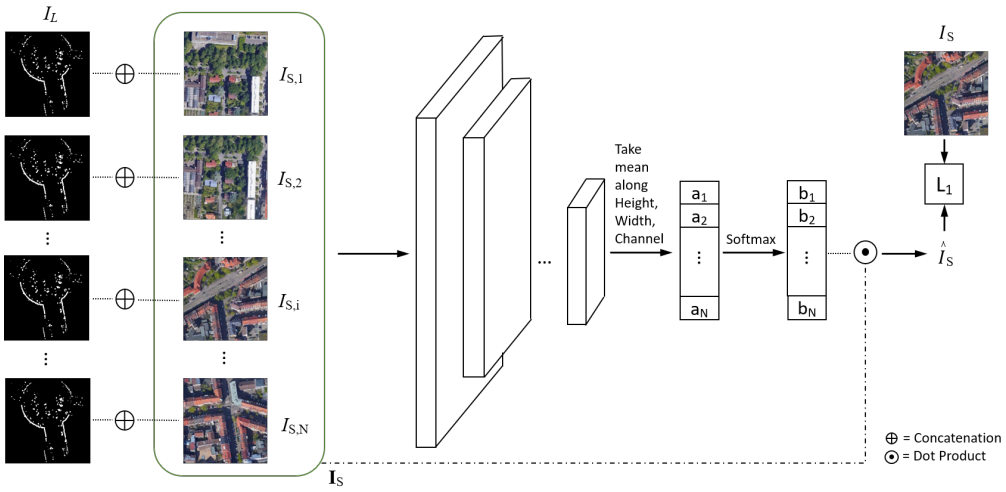


Fig. 12: Our ECML-CNN architecture. Birds-eye view (BEV) lidar images concatenated ($\oplus$) with an array of candidate satellite tiles serve as input to the model. The L_1 distance between the true satellite tile I_S and the generated tile $\hat{I}_S$ serves as the output energy of the system. $\hat{I}_S$ is constructed from the dot product $\mathbf{B} \cdot \mathbf{I}_S$ where the coefficients $\mathbf{B}$ are the softmax output of similarity scores $\mathbf{A}$ and the satellite array $\mathbf{I}_S$ is derived from $I_{S_{map}}$. At inference, we simply select the lidar-satellite pair that yields the highest similarity score a^* without using the softmax or the dot product.

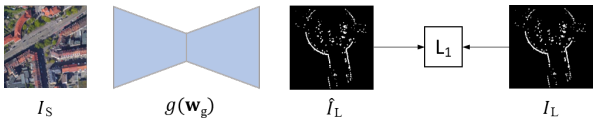


Fig. 13: The autoencoder used for the RSL baseline.

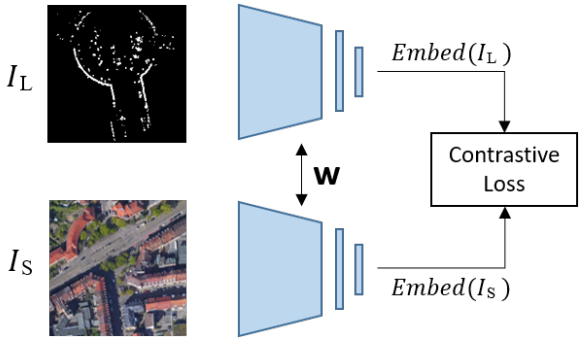


Fig. 14: The Siamese network used for one of the baselines.

is thus determined by the highest score achieved among all candidate lidar-satellite pairs:

$$(x^*, y^*, \theta^*) = \underset{x,y,\theta \in \mathcal{X},\mathcal{Y},\Theta}{\arg\max} f(\boldsymbol{I}_L(\theta), \boldsymbol{I}_S(x,y); \mathbf{w_f}) \quad (9)$$

While we search exhaustively in the translation space in the absence of GPS, we may limit our search space in the rotation space if we are provided a coarse heading estimate, such as from a noisy compass.

F. EBMs with Adversarial Learning

To estimate $f(\cdot; \mathbf{w_f})$, we train a convolutional transformer (ECML-CT-G) and a convolutional neural network (ECML-CNN-G) in a supervised manner to construct the satellite

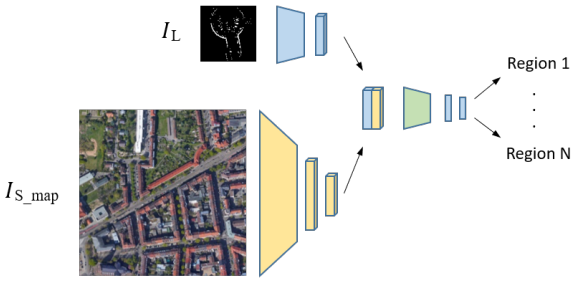


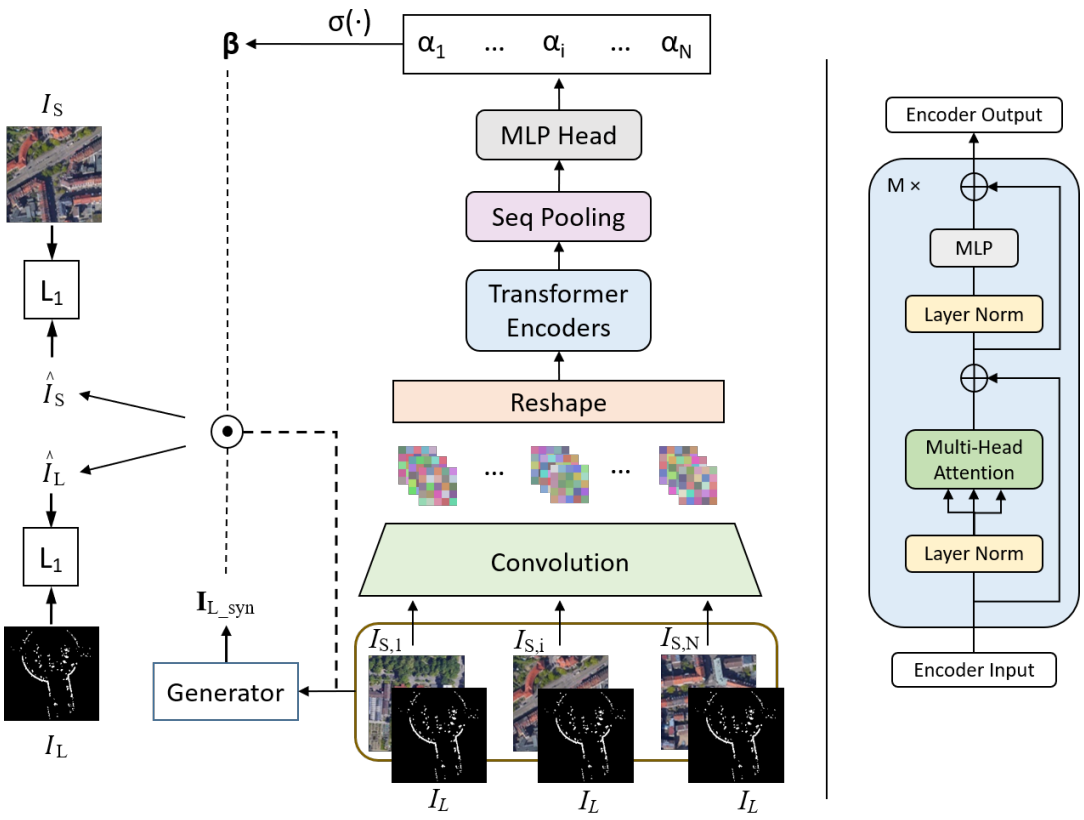
Fig. 15: The classification network used for one of the baselines.

tile $\hat{I}_S$ centered at the vehicle location from an ensemble of tile candidates. The probability of each candidate reflects the match quality between lidar-satellite image pairs, (I_L, I_{S_i}), where I_L is the aligned online lidar image and I_{S_i} is a satellite tile candidate derived from $I_{S_{map}}$, but not necessarily from a sliding window as during inference. We discuss in detail the composition of the input to the network used in training in section VI-A. The lidar-satellite image pairs, (I_L, I_{S_i}) are concatenated along the channel axis and serve as inputs to the CNN. To enhance performance, we add a second loss term that constructs the lidar image $\hat{I}_L$ from an ensemble of synthesized lidar images $\mathbf{I}_{L\_syn}$ that are generated from satellite images $\mathbf{I}_S$. We apply the L_1 loss between the predicted image and the target image for both modalities. Our objective is thus:

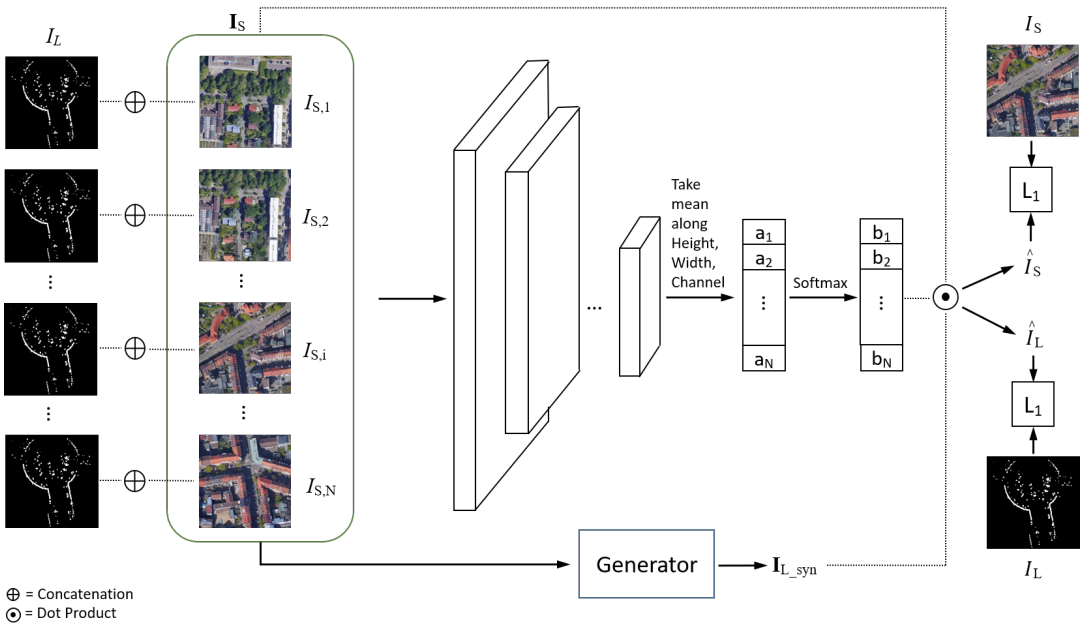
$$\mathcal{L}_f = \gamma \mathcal{L}_{L1}(\hat{I}_S, I_S) + \delta \mathcal{L}_{L1}(\hat{I}_L, I_L) \quad (10)$$

where $\gamma \in \{0, 1\}$ and $\delta \in \{0.1, 1, 10\}$ are weighting coefficients for the loss terms.

The CT shown in Fig. 16a shares the backbone described in Section III-C and the CNN in Fig. 16b shares the backbone described in Section VI-D, both attaining network output



(a) Energy-Based Model with CT backbone and adversarial learning: ECML-CT-G



(b) Energy-Based Model with CNN backbone and adversarial learning: ECML-CNN-G

Fig. 16: Alternative cross-modal localization networks with adversarial learning: ECML-CT-G (16a) and ECML-CNN-G (16b). Birds-eye view (BEV) lidar images concatenated ($\oplus$) with an array of candidate satellite tiles serve as input to the model. The L_1 distance between the true satellite tile I_S and the generated tile $\hat{I}_S$ in conjunction with the L_1 distance between the true lidar tile I_L and the generated tile $\hat{I}_L$ serves as the output energy of the system. $\hat{I}_S$ and $\hat{I}_L$ are constructed from the dot product $\mathbf{B} \cdot \mathbf{I}_S$ and $\mathbf{B} \cdot \mathbf{I}_{L\_syn}$ where the coefficients $\mathbf{B}$ are the softmax output of similarity scores $\mathbf{A}$, the satellite array $\mathbf{I}_S$ is derived from $I_{S_{map}}$, and the synthesized lidar array $\mathbf{I}_{L\_syn}$ is generated from $\mathbf{I}_S$. At inference, we simply select the lidar-satellite pair that yields the highest similarity score a^* without using the softmax or the dot product.

	e_x (pix)	e_y (pix)	e_θ (°)
KITTI			
ECML-CT	***4.7***	***8.2***	***2.12***
ECML-CT-G	7.6	11.2	2.84
ECML-CNN	12.2	14.8	3.53
ECML-CNN-G	11.8	17.2	3.18
PandaSet			
ECML-CT	8.8	***9.3***	***1.70***
ECML-CT-G	***7.2***	11.6	1.92
ECML-CNN	9.4	10.6	2.44
ECML-CNN-G	10.3	17.1	2.50
Custom			
ECML-CT	3.1	***1.6***	***1.69***
ECML-CT-G	***2.1***	1.9	1.96
ECML-CNN	8.0	6.4	2.58
ECML-CNN-G	11.6	5.4	3.26

TABLE III: Comparison of methods for mean distance error for x and y, and for mean angle error for θ on KITTI, Pandaset, and Custom A. Scale factor is 1.83 m/pixel.

$\mathbf{A} = [a_1, a_2, ...a_{N_S}]^T \in \mathbb{R}^{N_S}$, which we use as scores at inference. We apply a softmax operation to this vector to obtain a normalized probability distribution $\mathbf{B}$ such that

$$\sum_{i=1}^{N_S} b_i = 1 \tag{11}$$

The elements b_i serve as coefficients for a linear combination of satellite candidates $\mathbf{I}_S$ in constructing the predicted satellite image $\hat{I}_S = \mathbf{B} \cdot \mathbf{I}_S$. And for a linear combination of synthesized lidar images $\mathbf{I}_{L\_syn}$ in constructing the predicted lidar image $\hat{I}_L = \mathbf{B} \cdot \mathbf{I}_{L\_syn}$. Examples of generated satellite and lidar images can be found in Fig. 17.

Since we have access to the location of the vehicle at training, we can obtain the satellite image I_S centered at the vehicle location. Hence, our first loss term:

$$\mathcal{L}(\hat{I}_S, I_S) = ||I_S - \hat{I}_S||_1 \tag{12}$$

Naturally, we use the online lidar BEV image I_L as our target image for the second loss term:

$$\mathcal{L}(\hat{I}_L, I_L) = ||I_L - \hat{I}_L||_1 \tag{13}$$

To obtain $\mathbf{I}_{L\_syn}$, we learn a generator $G(\,\cdot\,; \mathbf{w_G})$ that translates satellite images into synthesized lidar images, where

$$G(\mathbf{I}_S; \mathbf{w_G}) \rightarrow \mathbf{I}_{L\_syn} \tag{14}$$

is governed by the parameters $\mathbf{w_G}$. The generator G is learned via the conditional generative adversarial network (CGAN) framework used by [66]. The objective for the CGAN is

$$G^* = \arg\min_G \max_D \mathcal{L}_{CGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{15}$$

where D is the discriminator and λ is a hyperparameter. The two loss terms $\mathcal{L}_{CGAN}$ and $\mathcal{L}_{L1}$ are expanded below:

$$\begin{aligned}\mathcal{L}_{CGAN}(G, D) = \mathbb{E}[\log D(I_S, I_{L_{syn}})] + \\ \mathbb{E}[\log (1 - D(I_S, G(I_S, z)))]\end{aligned} \tag{16}$$

Fig. 17: The odd columns contain generated satellite and lidar images, respectively. The even columns are the respective ground truth satellite and lidar images.

$$\mathcal{L}_{L1}(G) = \mathbb{E}[||I_L - G(I_S, z)||_1] \tag{17}$$

where z is Gaussian noise. At inference, we simply select the lidar-satellite pair that yields the highest similarity score a^* without using the softmax or the dot product.

G. Custom Dataset

For our custom dataset collected in the desert regions of southeastern California, we sometimes dealt with dust storms that degraded visibility. With the aid of lidar sensors, we were able to better sense our environment. Some samples are shown in Fig. 18.

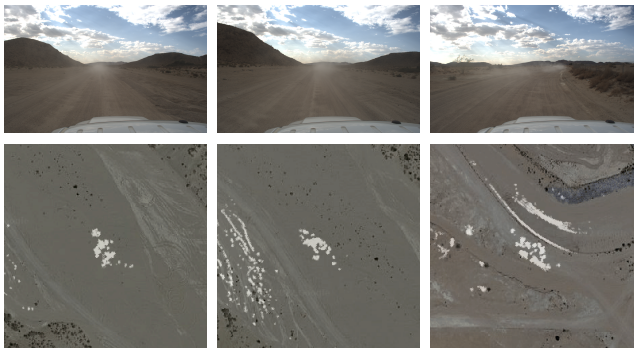


Fig. 18: Top row shows camera view. Bottom row shows lidar BEV overlaid on satellite imagery.

H. Satellite Imagery

Using onboard GPS data, we obtained the satellite imagery for all our datasets from Google Maps.