Self-Supervised Lidar Place Recognition in Overhead Imagery Using Unpaired Data

Tim Y. Tang, Daniele De Martini, and Paul Newman
Mobile Robotics Group, Oxford Robotics Institute, University of Oxford
Email: {ttang, daniele, pnewman}@robots.ox.ac.uk

Abstract—As much as place recognition is crucial for navigation, mapping and collecting training ground truth, namely sensor data pairs across different locations, are costly and time-consuming. This paper tackles these by learning lidar place recognition on public overhead imagery and in a self-supervised fashion, with no need for paired lidar and overhead imagery data. We learn the cross-modal data comparison between lidar and overhead imagery with a multi-step framework. First, images are transformed into synthetic lidar data and a latent projection is learned. Next, we discover pseudo pairs of lidar and satellite data from unpaired and asynchronous sequences, and use them for training a final embedding space projection in a cross-modality place recognition framework. We train and test our approach on real data from various environments and show performances approaching a supervised method using paired data.

I. INTRODUCTION

Lidar is widely considered an ideal sensor for outdoor operation, as it provides a long sensing range, 360\degree field-of-view (FOV), invariance to lighting, and robustness against weather conditions. For this reason, place recognition, also known as topological localisation, has been extensively researched for lidar [5, 13, 26, 18, 7, 38, 59]. Existing methods require lidar data to have been previously recorded in the places of operation, either as an aggregated point-cloud map or individual scans. When previously recorded sensory data are unreliable or unavailable, off-the-shelf overhead imagery, such as Google satellite images, can be used as an alternative map data source for place recognition. Even under normal operating conditions, overhead imagery can serve as an additional information source for redundancy.

When projected to the $x$-$y$ plane and expressed as a 2-D scan image, lidars capture geometric features also visible from bird’s-eye view aerial photos, providing useful signals for cross-comparison. Nevertheless, localising a lidar in a satellite image map remains challenging, as aerial imagery and range sensor scans are severely different. Recent works were proposed on pose estimation [54, 55] and place recognition [56] of radar and lidar using aerial images. Typically, learning place recognition requires paired data to form positive pairs in metric learning as in [56]. This, in turn, relies on accurate time-synced ground truth to query for a satellite image centred at the true centre position of each lidar scan, forming geometric one-to-one correspondences. In practice, collecting and post-processing time-synced GPS and inertial data for ground truth can require substantial cost and time. When live and map data are from the same sensor type, self-supervised learning with artificial paired data can be done by data augmentation [41, 16]. However, self-supervised metric learning is largely unsolved between cross-sensory data without known pairs.

We present a self-supervised method for learning lidar place recognition in overhead images from unpaired data, with an overview in Figure 1. Suppose a lidar-equipped vehicle travels along a known but never traversed route, we can, for example, sample satellite images uniformly along the route, forming a set of lidar and satellite data without geometric one-to-one correspondences (Figure 2, Middle). Our method learns from unpaired data, relaxing the need for on-board ground truth to collect time-synced, paired lidar and aerial data. Alternatively, if the route has been previously driven by a vehicle with an on-board GPS/INS, then asynchronous position measurements collected on GPS/INS timestamps from the prior drive can also be used to query for satellite images along the route (Figure 2, Right).
operate directly on point data or 2-D scan images. A. Deep Learning for Range Sensor Place Recognition

paired data when tested on unseen places. Our method is approaching a supervised approach trained with our method is approaching a supervised approach trained with unpaired data under the context of our problem set-up.

II. RELATED WORK

A. Deep Learning for Range Sensor Place Recognition

Neural networks for range sensor place recognition can operate directly on point data or 2-D scan images.

1) Point-based: Early work by Uy and Lee utilises NetVLAD after a PointNet backbone to learn a global descriptor for retrieval. A common strategy utilised by recent methods seeks to learn per-point local descriptors first and then aggregate them into a global descriptor for place recognition using pooling, normalisation, learned layers, or bag of words (BoW).

2) Image-based: Sun et al. expressed lidar data as bird’s-eye view images and directly learns a global pose, which is used to seed a Monte Carlo Localiser for pose refinement. Saftescu et al. expressed radar data as images in polar coordinates and learned rotation-invariant embeddings for place recognition via circular padding. Barnes and Posner predicted keypoints and pixel-level local descriptors from radar images and aggregated them to a per-image global descriptor via pooling for place recognition. OverlapNet uses 2-D range, normal, intensity, and semantic images extracted from 3D lidar data to predict the overlap between scans as a proxy for detecting loop closure. OverlapNet was extended to a Transformer-based architecture for rotation-invariant learning and to handle sequential data.

B. Localisation Using Aerial Imagery

Aerial imagery maps can be used to localise a forward-facing camera image in the geo-localisation problem. In this case, the localisation happens from different view perspectives (forward vs top-down). Still, the sensory nature of the query image and the map database are the same, i.e. RGB data. When a range sensor localises in an aerial imagery map, the view perspectives are consistent as range sensor scans are often expressed as bird’s-eye view images. However, the data belong to different modalities, requiring alternative strategies for bridging the sensory gap.

1) Cross-view, intra-modality: Early work by Lin et al. learns cross-view geo-localisation with hand-crafted features and an SVM classifier. CVMNet applies two streams of convolutional layers followed by a NetVLAD layer to learn cross-view matching with a weighted soft-margin ranking loss.

Li et al. tackled cross-view retrieval by supplying orientation maps, predicting cross-view orientation, learning spatial attention, and using optimal feature transport. Some methods extract hand-crafted features from overhead imagery before comparing them against range sensor data. Zhu et al. learned a matching probability between lidar grid-maps and satellite imagery to enhance a lidar SLAM pipeline. The work combines ground camera and lidar data to solve registration against overhead imagery in a correlation-maximisation approach. Tang et al. addressed the modality gap between range sensor data and satellite imagery by generating synthetic range sensor images or representing satellite imagery as points. Prior learning-based methods, different from our work, have mostly relied on paired range sensor and satellite data for supervision.

Most recently, Kim et al. trained a semantic segmentation network using hand-annotated satellite images and extracted features from OSM for global localisation. The methods use a hand-crafted descriptor for lidar and OSM data, and the modality gap between lidar and OSM by learning domain-invariant semantic descriptors. Cho et al. designed a hand-crafted rotation-invariant descriptor based on the distance to buildings applicable to both lidar point-clouds and OSM data.

C. Localisation Using Other Publicly Available Resources

Other off-the-shelf resources, in particular OSM, have been applied for robot localisation. The methods match Visual Odometry (VO) against road segments extracted from OSM for global localisation. The methods addressed the modality gap between lidar and OSM by learning domain-invariant semantic descriptors. Cho et al. designed a hand-crafted rotation-invariant descriptor based on the distance to buildings applicable to both lidar point-clouds and OSM data.

D. Cross-Sensory Retrieval With Unpaired Data

Several recent works have targeted cross-sensory retrieval with unpaired data. Yin et al. combined style transfer with joint feature space learning to match camera images with infra-red images. The method in uses a hand-crafted descriptor for lidar and OSM data, and therefore does not require paired data for training, but is not directly applicable to our problem set-up. The method in does not require paired lidar and satellite data, but relies on hand-annotated satellite imagery semantics to train semantic segmentation, which is another form of ground truth that is potentially time consuming to acquire.

Though promising, the method by Yin et al. seeks to address the modality gap between two types of range sensors, while the method in targets RGB images and infrared images, both having a much smaller modality gap than between lidar and aerial images. In our experiments, neither nor were sufficient in our problem of lidar place recognition in satellite imagery with unpaired data.
E. Unpaired Image-to-Image Translation

A core module of our pipeline involves generating synthetic lidar images from satellite images in an unpaired set-up. Unpaired image-to-image translation can be achieved using cycle-consistency \cite{8} between the input and reconstructed images, the assumption of a shared latent space \cite{33,22}, or contrastive learning \cite{44}. Several prior works were proposed to learn multi-modal image generation \cite{22,30,1}, but empirically they resulted in limited success in generating lidar images from satellite images in our experiments. Our choice fell on CycleGAN \cite{64} as it is a well-established method.

III. PROBLEM OVERVIEW

Suppose a vehicle travels along a known route and takes lidar scans at a certain frequency, resulting in a sequence of \( N \) lidar images \( \mathcal{X} = \{ X_1, X_2, \ldots, X_N \} \). Satellite images are queried along the route, forming a sequence of \( M \) satellite images \( \mathcal{Y} = \{ Y_1, Y_2, \ldots, Y_M \} \). In the simplest case, paired data are available through ground truth: for each lidar scan \( X_i \), we have a satellite image \( Y_j \) sharing the same centre position, resulting in one-to-one correspondences with \( N = M \).

Here, we consider a more general scenario where paired data are unavailable and the lidar and satellite sampling positions differ, for example, if the satellite images are sampled uniformly along the route. In this context, we want to find for each \( X_i \), the closest \( Y_j \in \mathcal{Y} \); vitally, there are no known correspondences from training data, and the mapping from \( \mathcal{X} \) to \( \mathcal{Y} \) is neither surjective nor injective.

While this is trivial if \( \mathcal{X} \) and \( \mathcal{Y} \) are paired, our method seeks to extract positive pairs from unpaired data.

IV. METHODOLOGY

The core idea of our approach is to exploit the fact that, albeit without one-to-one correspondences, lidar and satellite data follow the same underlying sequence, as they are collected along the same known route. We begin by learning an embedding function for projecting a lidar image to a vector space descriptor using time consistency on lidar only (Section IV-B). In parallel, we learn to generate synthetic lidar images from satellite imagery using CycleGAN \cite{64} (Section IV-C). We can then use this projection to construct an affinity matrix by comparing synthetic and real lidar images.

Though the synthetic lidar images are realistic, there is no guarantee they capture the same regions of the scene as a real lidar situated at the centre of the satellite image. The affinity matrix will then be corrupted by heavy signal aliasing. We propose a simple yet effective learned strategy to de-alias the affinity matrix and improve its signal-to-noise ratio before sequence alignment (Section IV-D). Pseudo pairs are then selected from sequence alignment and used as positive pairs to learn a final embedding space projection for place recognition (Section IV-E). Figure 1 shows the method overview.

A. Data Representation

In our problem set-up, we project 3D lidar data to the \( x-y \) plane to form bird’s-eye view lidar images, where points with \( z \) value less than a threshold are discarded to remove ground points. The intensity in each pixel is the average intensity of all points projected to that pixel.

Since the heading offset between lidar scans and satellite data is unknown, the place recognition pipeline must be rotation-invariant. Motivated by this, we convert lidar and satellite images to a polar coordinate representation, where the axes are range \( r \) and azimuth \( \theta \). A rotation in Cartesian space becomes a circular shift in the polar domain along the vertical axis, and, as CNNs are equivariant to vertical shifts, they can be easily trained to be rotation-invariant through data augmentation when applied to polar images. Examples of polar representations of satellite and lidar images are shown in Figure 4. While recent work proposes the Radon transform, which is \( SE(2) \) equivariant, for lidar localisation \cite{37}, we observed that Radon transform fails to preserve regions in satellite images with a strong gradient, such as building edges.

B. Descriptor Embedding for Lidar Images

We wish to learn a function \( F \), parametrised by a neural network, that projects a lidar image to a descriptor space
\( \mathbb{R}^d \), where closeness in Euclidean space reflects closeness in Cartesian space geometrically. We train the function \( F \) in a self-supervised way using Siamese networks with a triplet margin loss. Specifically, given a polar lidar image \( X_i \), we take its temporal neighbour taken \( \tau \) frames earlier or later, \( X_{i\pm\tau} \), to form a positive pair, and take a random sample \( X^- \) to form a negative pair, and minimise the following loss:

\[
\mathcal{L}_{\text{emb}} = \frac{1}{m} \left( \frac{1}{2} \left[ \| F(X_i) - F(X_{i\pm\tau}) \| - \| F(X_i) - F(X^-) \| + m \right] \right).
\]

\([a]_+ \) here denotes \( \max(a, 0) \), and \( m \) is the triplet margin. In our experiments, we set \( m \) to 1 and \( \tau \) to 5. We apply random rotation augmentations to \( X_i, X_{i\pm\tau} \), and \( X^- \) so that \( F \) learns to be rotation invariant.

### C. Unpaired Satellite-to-Lidar Translation

To bridge the modality gap between satellite imagery and lidar, we use a variant of CycleGAN applied to the polar images. Specifically, we concatenate each polar lidar or satellite image with an additional single-channel image \( R \), where the values of all pixels on column \( k \) of \( R \) is \( \frac{R}{255} \), with \( W \) being the image width. Since height and width now denote azimuth and range values in polar representation, supplying \( R \) makes the generator and discriminator aware of the normalised range value of each pixel, and has shown to help stabilise training.

In CycleGAN, we seek to optimise generator functions \( G_{X \rightarrow Y}, G_{Y \rightarrow X} \) and discriminator functions \( D_X, D_Y \), parametrised by neural networks. \( G_{X \rightarrow Y} \) maps an image from modality \( X \) to \( Y \), while \( G_{Y \rightarrow X} \) is the counterpart. \( D_X \) and \( D_Y \) discriminate whether an image in modality \( X \) or \( Y \) is real or fake, respectively.

Given arbitrary \( X_i \) and \( Y_j \), an adversarial loss can be applied to \( G_{X \rightarrow Y} \) and \( D_Y \):

\[
\mathcal{L}_{\text{GAN}}(G_{X \rightarrow Y}, D_Y, X_i, Y_j, R) = \mathbb{E}_{Y_j \sim Y} \log D_Y(Y_j, R)
+ \mathbb{E}_{X_i \sim X} \log \left( 1 - D_Y(G_{X \rightarrow Y}(X_i, R), R) \right),
\]

and a similar loss is introduced for \( G_{Y \rightarrow X} \) and \( D_X \).

The cycle-consistency loss can be formulated as:

\[
\mathcal{L}_{\text{cyc}} = \mathbb{E}_{X_i \sim X} \left\| X_i - G_{Y \rightarrow X}(G_{X \rightarrow Y}(X_i, R), R) \right\|_1
+ \mathbb{E}_{Y_j \sim Y} \left\| Y_j - G_{X \rightarrow Y}(G_{Y \rightarrow X}(Y_j, R), R) \right\|_1.
\]

Finally, the full loss is:

\[
\mathcal{L}_{\text{CycleGAN}} = \mathcal{L}_{\text{GAN}}(G_{X \rightarrow Y}, D_Y, X_i, Y_j, R)
+ \mathcal{L}_{\text{GAN}}(G_{Y \rightarrow X}, D_X, Y_j, X_i, R) + \lambda \mathcal{L}_{\text{cyc}},
\]

where \( \lambda \) was set to 100 in our experiments. The networks are optimised as:

\[
G^*_X \rightarrow Y, G^*_Y \rightarrow X = \arg\min_{G_{X \rightarrow Y}, D_Y} \arg\max_{G_{Y \rightarrow X}, D_X} \mathcal{L}_{\text{CycleGAN}}.
\]

As the majority of pixels in a lidar image are dark and only a small fraction has range returns, there is no guarantee synthetic lidar images will capture the same regions of the scene as an actual lidar situated at the centre of the satellite image would. Figure 5 shows examples of synthetic lidar images compared to actual lidar images taken at the centre of the satellite image (not used during training in our method). In many cases the synthetic lidar images are visibly significantly different from what an actual lidar would capture, which will result in finding false matches in the descriptor space. As such, relying on unpaired image-to-image translation only is insufficient, unlike the case of radar-to-lidar transfer where locations with strong radar range return will likely also result in lidar return, and pixels with weak or no radar return will likely be not captured in a lidar scan.

### D. Sequence Alignment

Though individual nearest-neighbour matching with synthetic lidar images is inadequate, pair-finding can be achieved with sequence alignment. Specifically, given a sequence of lidar images \( X = \{X_1, X_2, \ldots, X_N\} \) and a sequence of satellite images \( Y = \{Y_1, Y_2, \ldots, Y_M\} \) queried along the same route, we first generate synthetic lidar images \( \tilde{X}_j = G_{Y \rightarrow X}(Y_j) \) using the learned generator, forming a set of synthetic lidar images \( \tilde{X} = \{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_M\} \). Next, we can
construct an affinity matrix \( A \in \mathbb{R}^{N \times M} \) where each element is the Euclidean distance in descriptor space between \( X_i \) and \( \tilde{X}_j \) after normalisation:

\[
\begin{align*}
    f_i &= \frac{F(X_i) - \mu_X}{\sigma_X}, \\
    \tilde{f}_j &= \frac{F(\tilde{X}_j) - \mu_{\tilde{X}}}{\sigma_{\tilde{X}}}, \\
    A_{ij} &= \|f_i - \tilde{f}_j\|, 
\end{align*}
\]

where \( F \) is the learned embedding function from Section IV-B, \( \mu_X \in \mathbb{R}^d \) and \( \sigma_X \in \mathbb{R}^d \) are the mean and standard deviation of \( \{F(X_1), \ldots, F(X_N)\} \), and similarly for \( \mu_{\tilde{X}} \) and \( \sigma_{\tilde{X}} \).

Since synthetic lidar images are not necessarily compatible with real lidar images as described in Section IV-C, the affinity matrix can be heavily corrupted by signal aliasing, resulting in a poor signal-to-noise ratio. Figure 6a shows the affinity matrix for a sequence from the KITTI dataset [17]. Various hand-crafted methods were proposed in prior work to enhance the contrast in the affinity matrix. SeqSLAM [42] proposes 1-D patch normalisation on the affinity matrix. Ho and Newman [20] performed eigendecomposition on the affinity matrix and reconstructed a rank-reduced one by removing eigenvectors corresponding to the largest eigenvalues. The methods in [48, 34] apply dimensionality reduction by keeping the descriptor’s most descriptive \( k \leq d \) dimensions. These methods were designed primarily for visual place recognition – where the mapping and localising sensors are of the same type and thus the affinity matrix does not suffer from extreme signal aliasing as in our cross-sensory problem.

We propose **neural de-aliasing**, a learned approach for de-aliasing the affinity matrix trained on simulated data. First, we form \( K \) random vectors \( \{\xi_1, \xi_2, \ldots, \xi_K\} \), where each \( \xi_k \in \mathbb{R}^d \) is sampled from a zero-mean Gaussian distribution, and normalised to a unit sphere. Next, we form a random dynamic sequence of length \( P \), \( \Phi = \{\phi_1, \ldots, \phi_P\} \), by travelling from \( \xi_1 \) to \( \xi_K \), taking step sizes of 0, 1, or 2 with various probabilities each. This procedure is repeated, forming a different dynamic sequence of length \( Q \), \( \Psi = \{\psi_1, \ldots, \psi_Q\} \). We can construct an affinity matrix \( A_s \in \mathbb{R}^{P \times Q} \) between \( \Phi \) and \( \Psi \). Here, \( P \) and \( Q \) can either be less than, equal to, or larger than \( K \). The probability for each step size is a design choice and does not affect performance greatly.

We then add artificial aliasing to corrupt the simulated affinity matrix. We take inspiration from [20], yet, instead of removing eigenvectors, we introduce aliasing by adding an eigenvector from the affinity matrix of a KITTI sequence whose real and synthetic lidar embeddings result in heavy aliasing. The detailed procedure is found in Algorithm 1 and a visual example of a simulated affinity matrix \( A_s \) and its alias-corrupted version \( A'_s \) is shown in Figure 7.

We can treat \( A_s \) and \( A'_s \) as single-channel images and apply Pix2Pix [23] to learn to recover a clean affinity matrix from its alias-corrupted counterpart, after appropriate resizing:

\[
A_s = G_{DA}(A'_s),
\]

where \( G_{DA} : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{H \times W} \) is a matrix de-aliasing function parametrised by a neural network. After \( G_{DA} \) is
Algorithm 1: Adding Signal Aliasing

Input:
\( A_s \in \mathbb{R}^{P \times Q} \) # affinity matrix from simulated data 
\{\hat{f}_1, \ldots, \hat{f}_N\}, \{\tilde{f}_1, \ldots, \tilde{f}_M\} # embeddings from KITTI

Output:
\( \Lambda_A' \) # affinity matrix with added signal aliasing

Procedure:
\( f \in \mathbb{R}^{d \times (N+M)} \leftarrow [f_1 \ldots f_N \ \hat{f}_1 \ldots \hat{f}_M] \)
\( W \in \mathbb{R}^{(N+M) \times (N+M)} \leftarrow \text{initialise} \)
for \( i = 1, 2, \ldots, N+M \) do
\hspace{1cm} for \( j = 1, 2, \ldots, N+M \) do
\hspace{2cm} \# \( f_i \) and \( f_j \) are the \( i^{th} \) and \( j^{th} \) columns of \( f \)
\hspace{2cm} \( W_{ij} = \| f_i - f_j \| \)
\hspace{1cm} \( \Lambda, V \leftarrow \text{eigendecomposition}(W) \)
\hspace{1cm} \( \lambda^*, v^* \leftarrow \text{largest eigenvalue and the corresponding eigenvector} \)
\hspace{1cm} \( v^* \leftarrow \text{randompermute}(v^*) \)
\hspace{1cm} \( N \in \mathbb{R}^{(N+M) \times (N+M)} \leftarrow v^* \lambda^* v^{*\text{T}} \)
\hspace{1cm} # random crop to a \( P \times Q \) patch
\hspace{1cm} \( N \in \mathbb{R}^{P \times Q} \leftarrow \text{randomcrop}(N, P, Q) \)
\hspace{1cm} \( A_s' \leftarrow A_s + N \)
\hspace{1cm} \( A_s' \leftarrow A_s'/\max(A_s') \)

Fig. 7: Example of a simulated affinity matrix \( A_s \) (left) and its alias-corrupted version \( A_s' \) (right). We train a neural network that takes \( A_s' \) as input and recovers the original, high-contrast affinity matrix \( A_s \).

After pseudo pairs are selected from sequence alignment, we use them as positive pairs in metric learning for place recognition. Formally, we wish to learn embedding functions \( F_X \) and \( F_Y \) that project images of modality \( \mathcal{X} \) and \( \mathcal{Y} \) respectively to \( \mathbb{R}^{D} \), where \( F_X \) and \( F_Y \) are parametrised by neural networks. Here \( F_X \) and \( F_Y \) are separate and different from \( F \) as in Section [IV-B] which was used for constructing the affinity matrix. To optimise \( F_X \) and \( F_Y \), we aim to minimise a bi-directional triplet margin loss:

\[
\mathcal{L}_{PR} = \left[ \| F_X(X_p) - F_Y(Y_q) \| - \| F_X(X_p) - F_Y(Y^-) \| + m \right]_+
\]

\[
+ \left[ \| F_Y(Y_q) - F_X(X_p) \| - \| F_Y(Y_q) - F_X(X^-) \| + m \right]_+,
\]

where \((X_p, Y_q) \in S\), and the negative samples \( X^- \) and \( Y^- \) are random samples from \( \mathcal{X} \) and \( \mathcal{Y} \). Here, we set the triplet margin \( m \) to 0.1. We again apply random rotation augmentations to the images, so \( F_X \) and \( F_Y \) learn to be rotation-invariant.

F. Network Architecture and Implementation Details

For the generator networks \( G_{X \rightarrow Y}, G_{Y \rightarrow X} \), and \( G_{DA} \), we use the ResNet [19] generator from the authors’ official implementation [6]. For learning embeddings, we use the convolution layers of a VGG16-style [52] backbone followed by a NetVLAD layer [2] for \( F \), and the convolution layers of a ResNet18 backbone, followed by a NetVLAD layer for \( F_X \) and \( F_Y \). On lidar and synthetic lidar images, we add a single convolution layer at the top to convert single-channel images to 3 channels, so VGG16 and ResNet18 can consume them. We set the embedding dimensions as \( d = 256 \) and \( D = 2048 \).

All of our training is conducted in PyTorch with a batch size of 32. We use RMSProp with a fixed learning rate of \( 1 \times 10^{-4} \) in CycleGAN training and ADAM with a fixed learning rate of \( 2 \times 10^{-4} \) in all other modules. As no ground truth data are used in training, we cannot split the training data to form a validation set. Instead, we arbitrarily terminate training when the training loss has stabilised for 10 epochs.

V. EXPERIMENTAL SETUP

Our method is validated on the Oxford Radar RobotCar Dataset [4], of which we use the left of the two Velodyne HDL-32E lidars mounted in a tilted configuration, and the KITTI Dataset (raw data) [17], which uses a Velodyne HDL-64E lidar mounted on-top.

Our experiments’ trajectories at inference time are unseen during training. As RobotCar features repeated sequences of the same route, we split the trajectory into training and test sets with no overlap, as shown in Figure 9a. We use lidar data collected along the training trajectories from sequences no. 2 and 5 and query satellite data according to the GPS/INS timestamps of sequence no. 7, forming an unpaired training set illustrated in Figure 2 Right.

On KITTI, the training set consists of two long residential sequences, \( 2011_10_03_0034 \) and \( 2011_10_03_0034 \). To simulate the scenario where the route is known from a past survey (but never traversed by the lidar-equipped vehicle), we take the paired GPS data but uniformly divide each sequence into 5000 segments based on distance and find the centre of each segment. We then sample satellite images at these centres (as in Figure 2 Middle) with an added random error of –2 m to 2 m to simulate the past survey following a slightly different

https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
route than when collecting lidar data. The test set consists of another long residential sequence, 2011_09_30_0028, where we discard the first 1000 frames to avoid overlap with training data, shown in Figure 9b.

![Fig. 8: Results of applying neural de-aliasing compared to hand-crafted techniques on the affinity matrix of sequence no.2 from RobotCar.](image)

(a) Original affinity matrix  (b) 1-D patch normalisation  (c) Eigenvector removal  (d) Ours (neural de-aliasing)

Learning lidar-only place recognition in satellite imagery is relatively unstudied, and we compare against the following baselines, whose training data requirements are summarised in Table I.

**Paired baseline:** We trained the same network as ours with the same metric learning approach as in Section IV-B, except with paired training data. Although the paired lidar and satellite images do not need to be perfectly aligned at pixel level, for example the heading ground truth is not required.

**Unpaired baseline:** This baseline method uses the descriptor embedding function $F$ trained on real lidar images only as in Section IV-B and directly operates on synthetic lidar images generated from satellite images using the approach in Section IV-C. Specifically, for the unpaired baseline, all test set satellite images are mapped to descriptor space as $\{F(G_Y \rightarrow_X(Y_j))\}$, and each lidar image at test time is mapped to descriptor space as $F(X_i)$. The unpaired baseline is similar in spirit as [62] but applied to lidar-satellite place recognition and makes no attempt at sequence alignment.

**PointLoc:** To the best of our efforts, we implement [56] where satellite images are converted to 2-D points for comparison against lidar data, which we dub PointLoc. PointLoc requires the data to be paired and fully aligned at pixel-level through accurate $SE(2)$ ground truth, including heading.

**VI. EXPERIMENTAL RESULTS**

**A. RobotCar Dataset**

The affinity matrix for sequence no.2 of RobotCar is shown in Figure 8a. It has much less signal aliasing than the KITTI dataset (Figure 6a). Though hand-crafted techniques such as 1-D patch normalisation and eigenvector removal can increase local contrast to a sufficient extent, we show qualitatively in Figure 8a that our learned method reduces aliasing even further.

The test set trajectory of the RobotCar dataset features approximately 1 km of urban environment, with 800 lidar frames sampled at 4 Hz. The retrieval performances of our method compared to the baselines are shown in Table I. Though slightly outperformed by the supervised, paired method, almost half of our top-1 retrievals can localise correctly within 60 m of the true position. Figure 10 shows the Precision-Recall curve, where our method has higher precision than using paired data for recall between 2% and 10%. PointLoc

<table>
<thead>
<tr>
<th>Paired data</th>
<th>Metric $SE(2)$</th>
<th>Ours</th>
<th>Paired baseline</th>
<th>Unpaired baseline</th>
<th>PointLoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground truth</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\times$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>
greatly outperforms the other methods, but has the strictest requirements for training data ground truth.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired baseline</td>
<td>3.08</td>
<td>3.69</td>
<td>4.91</td>
<td>6.15</td>
<td>7.42</td>
<td>8.41</td>
</tr>
<tr>
<td>Unpaired baseline</td>
<td>8.25</td>
<td>17.00</td>
<td>19.63</td>
<td>20.75</td>
<td>21.75</td>
<td>22.50</td>
</tr>
<tr>
<td>PointLoc [56]</td>
<td>48.00</td>
<td>57.00</td>
<td>60.63</td>
<td>63.75</td>
<td>66.50</td>
<td>68.88</td>
</tr>
<tr>
<td>Ours</td>
<td>17.00</td>
<td>32.75</td>
<td>37.13</td>
<td>41.25</td>
<td>43.63</td>
<td>45.50</td>
</tr>
<tr>
<td>Top 1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired baseline</td>
<td>29.91</td>
<td>42.70</td>
<td>47.52</td>
<td>49.21</td>
<td>50.77</td>
<td>52.98</td>
</tr>
<tr>
<td>Unpaired baseline</td>
<td>6.84</td>
<td>12.55</td>
<td>14.58</td>
<td>16.21</td>
<td>17.31</td>
<td>18.44</td>
</tr>
<tr>
<td>PointLoc [56]</td>
<td>43.14</td>
<td>54.15</td>
<td>59.29</td>
<td>63.22</td>
<td>66.46</td>
<td>69.33</td>
</tr>
<tr>
<td>Ours</td>
<td>20.05</td>
<td>29.93</td>
<td>34.83</td>
<td>38.74</td>
<td>41.08</td>
<td>43.20</td>
</tr>
</tbody>
</table>

**TABLE II:** Percentage of top-1 and top 1% retrievals within each distance threshold to the true position, evaluated on the RobotCar Dataset.

![Fig. 10: Precision-Recall curve on the test set of RobotCar.](image1)

**B. KITTI Dataset**

The test set trajectory of KITTI features approximately 4 km of traversal with more than 4000 lidar frames at 10 Hz. KITTI features much more challenging data than RobotCar for lidar place recognition using overhead imagery. Firstly, KITTI uses a Velodyne HDL-64E lidar compared to Velodyne HDL-32E. Though the higher vertical resolution is ideal for many lidar-based applications, it increases complexity in the resulting lidar images, making it less likely for the synthetic lidar images to be compatible with real lidar scans. This has made it extremely challenging to perform sequence alignment from unpaired data, as indicated by the significantly higher levels of aliasing (Figure 6a) than RobotCar (Figure 8a). Moreover, the test set is in a residential area with many similar places and fewer distinct landmarks, resulting in a high false positive rate. This is further exaggerated when satellite images are expressed as points where several distinctive image features are lost, as demonstrated by the reduced performance of PointLoc. Finally, our test set of KITTI features a longer trajectory, as demonstrated by the reduced performance of PointLoc.

![Fig. 11: Precision-Recall curve on the test set of KITTI.](image2)

**C. Monte Carlo Localisation**

Though single frame retrieval using only overhead imagery has limited accuracy, given a stream of lidar data, here we present a Monte Carlo Localisation (MCL) pipeline that can accurately track the pose over long distances. Assuming the route to be taken at test time is known, we can formulate the localisation as a 1-D problem along the known route, where the distance along the route parameterises the state.

Specifically, at a time $t$, the state consists of $J$ particles $P^t = \{p^t_1, \ldots, p^t_J\}$ denoting the distance from the starting point of the trajectory, and the 1-D parametrisation indexes to a 2-D $x$-$y$ position in the world. We use 2000 particles in our experiments, uniformly initialised along the trajectory. The detailed MCL update step is shown in Algorithm 2.

1) **Motion Model:** A crucial step in MCL is the motion update, typically provided by vehicle control input or odometry. To demonstrate a pipeline that uses only lidar place recognition in overhead imagery with no need for accurate odometry, we update the motion by sampling the velocity from a uniform mean Gaussian distribution with mean $\mu_v$ and standard deviation $\sigma_v$. We choose $\mu_v$ based on prior knowledge about the vehicle’s speed. For the RobotCar Dataset, we set $\mu_v$ to 20 km/h as each sequence features approximately a 10 km trajectory collected in around 30 minutes. For KITTI, there is no repeated traversal, so we set $\mu_v$ to 30 km/h, which is the speed limit in residential areas in Germany. Finally, as the motion update is entirely approximated based on a constant-speed prior, which may not represent the vehicle’s true motion at time $t$, we use a large value of $\sigma_v$ corresponding to 10 m/s for both datasets to account for the noise correctly.

2) **Measurement Model:** For each particle $p^t_j$, its associated $x$-$y$ position is used to query the nearest satellite image. The corresponding satellite image is mapped to descriptor space using $F_Y$, and compared to the live lidar image at time $t$, $X^t$.

3) **Results:** We compute the median of $P^t$ to find the vehicle’s estimated distance along the trajectory, and thus its $x$-$y$ position, at time $t$. The error to the ground truth position is plotted for RobotCar in Figure 12 and KITTI in Figure 13.

**TABLE III:** Percentage of top-1 and top 1% retrievals within each distance threshold to the true position, evaluated on the KITTI Dataset.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired baseline</td>
<td>3.42</td>
<td>5.89</td>
<td>7.90</td>
<td>10.61</td>
<td>12.09</td>
<td>13.31</td>
</tr>
<tr>
<td>Unpaired baseline</td>
<td>1.41</td>
<td>2.18</td>
<td>3.02</td>
<td>3.38</td>
<td>4.00</td>
<td>5.41</td>
</tr>
<tr>
<td>PointLoc [56]</td>
<td>3.08</td>
<td>3.69</td>
<td>4.91</td>
<td>6.15</td>
<td>7.42</td>
<td>8.41</td>
</tr>
<tr>
<td>Ours</td>
<td>2.61</td>
<td>4.93</td>
<td>7.02</td>
<td>8.93</td>
<td>9.87</td>
<td>10.92</td>
</tr>
</tbody>
</table>

**Paired baseline**

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired baseline</td>
<td>3.58</td>
<td>6.34</td>
<td>8.39</td>
<td>10.34</td>
<td>11.83</td>
<td>13.12</td>
</tr>
<tr>
<td>Unpaired baseline</td>
<td>2.17</td>
<td>1.96</td>
<td>2.72</td>
<td>3.46</td>
<td>4.22</td>
<td>4.89</td>
</tr>
<tr>
<td>PointLoc [56]</td>
<td>2.68</td>
<td>3.93</td>
<td>5.08</td>
<td>6.18</td>
<td>7.35</td>
<td>8.48</td>
</tr>
<tr>
<td>Ours</td>
<td>1.89</td>
<td>3.83</td>
<td>5.43</td>
<td>6.92</td>
<td>8.04</td>
<td>9.01</td>
</tr>
</tbody>
</table>
Algorithm 2: 1-D Monte Carlo Localisation

\textbf{function} MCL(\(P^{t-1}_{j}, \mu_v, \sigma_v, X_j\))

\[ P^t = P^{t-1}_j = \emptyset \]

for \( j = 1, 2, \ldots, J \) do

\begin{align*}
& \text{sample } v \sim \mathcal{N}(\mu_v, \sigma_v) \\
& p^j_t \leftarrow p^j_{t-1} + v \cdot \Delta t \\
& Y^j_t \leftarrow \text{nearest satellite image to } p^j_t \\
& s^j_t \leftarrow \frac{\|F_X(X^j)\|-\|F_Y(Y^j)\|}{\|F_X(X^j)\| + \|F_Y(Y^j)\|} \quad \# \text{cosine similarity} \\
& P^t \leftarrow P^t \oplus p^j_t \\
& S^t \leftarrow S^t \oplus s^j_t \\
\end{align*}

return \(P^t\)

for our method and the paired baseline. On RobotCar, the particles quickly converged after about 50 frames and localised with small errors afterwards. On KITTI, though MCL lost track near the end of the trajectory, it successfully localised to mostly under 50 m error for over 2 km, relying solely on comparing lidar data against overhead imagery. Notably, the localisation accuracy of our method is on par with the paired baseline under an MCL set up.

![MCL results on RobotCar](image1)

**Fig. 12:** MCL results on RobotCar.

![MCL results on KITTI](image2)

**Fig. 13:** MCL results on KITTI.

D. Unpaired Radar Place Recognition in Overhead Imagery

Our method was designed for lidar data but can be applied to radar data through radar-to-lidar image translation. Given radar images \(Z = \{Z_1, Z_2, \ldots\}\), we apply CycleGAN [64] to learn a function \(G_{Z \rightarrow X} : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{H \times W}\) that maps a polar radar image to its synthetic lidar counterpart. Figure [14] shows examples of radar images and the corresponding synthetic lidar images after style transfer on the RobotCar dataset, which also features an on-board Navtech radar.

![Radar images (yellow) and corresponding synthetic lidar images](image3)

**Fig. 14:** Radar images (yellow) and corresponding synthetic lidar images.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired baseline</td>
<td>11.25</td>
<td>16.25</td>
<td>22.88</td>
<td>30.88</td>
<td>36.88</td>
<td>46.13</td>
</tr>
<tr>
<td>Ours</td>
<td>9.75</td>
<td>22.38</td>
<td>29.75</td>
<td>36.25</td>
<td>41.50</td>
<td>45.50</td>
</tr>
<tr>
<td>Top 1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paired baseline</td>
<td>10.70</td>
<td>17.83</td>
<td>23.95</td>
<td>31.47</td>
<td>37.80</td>
<td>45.54</td>
</tr>
<tr>
<td>Ours</td>
<td>9.26</td>
<td>22.12</td>
<td>30.29</td>
<td>36.02</td>
<td>41.32</td>
<td>45.72</td>
</tr>
</tbody>
</table>

**TABLE IV:** Percentage of top-1 and top 1% retrievals within each distance threshold to the true position, evaluated on RobotCar for radar data.

Then, without further retraining, we utilise networks learned with lidar data from our unpaired approach and directly apply them to synthetic lidar images from radar input. Specifically, taking \(F_X, F_Y\) trained for lidar-satellite place recognition with found pseudo pairs, at test time, each radar image \(Z_i\) is mapped to descriptor space as \(F_X(G_{Z \rightarrow X}(Z_i))\), and compared against the map database of satellite image descriptors \(\{F_Y(Y_j)\}\). We compare our approach against a supervised method trained on paired radar and satellite images, using the metric learning approach in Section IV-E. The retrieval performance is shown in Table [IV]. Notably, the unpaired approach has better accuracy than the supervised approach trained with paired radar and satellite data.

VII. CONCLUSION AND FUTURE WORK

In this paper, we show how lidar place recognition in a map of satellite images can be solved without any paired data. Our approach relaxes the need for an on-board, time-synchronised GPS/INS when collecting on-road lidar data for high-precision pose ground truth, as long as the route taken is known from a previous survey or traversal. Though the performance of place recognition in overhead imagery is far from lidar-to-lidar localisation, this capability nevertheless allows a mobile robot to travel to an unvisited place and still localise to a certain extent, and can also add an extra layer of redundancy in standard navigation applications.

Here we focus on solving the pairing problem from unpaired data and rely on metric learning for solving place recognition. Using data with global pose ground truth, future work can target how to further bridge the domain gap between lidar data and overhead imagery to enhance the localisation quality.
REFERENCES


[24] Somi Jeong, Seungryong Kim, Kihong Park, and


