2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636231

Trajectory Generation in New Environments from Past Experiences

Weiming Zhi[1,*], Tin Lai[1,*], Lionel Ott[2], Fabio Ramos[1,3]

***Abstract*—Being able to safely operate for extended periods of time in dynamic environments is a critical capability for autonomous systems. This generally involves the prediction and understanding of motion patterns of dynamic entities, such as vehicles and people, in the surroundings. Many motion prediction methods in the literature implicitly account for environmental factors by learning on observed motion in a fixed environment, and are designed to make predictions in the same environment. In this paper, we address the problem of generating likely motion trajectories for novel environments, represented as occupancy grid maps, where motion has not been observed. We introduce the Occupancy-Conditional Trajectory Network (OTNet) framework, capable of transferring the previously observed motion patterns in known environments to new environments. OTNet provides a functional representation for motion trajectories and utilises neural networks to learn occupancy-conditional distributions over the function parameters. We empirically demonstrate our method's ability to generate complex multi-modal trajectory patterns in both simulated and real-world environments.**

I. Introduction

Understanding movement trends in dynamic environments is critical for autonomous agents, such as service robots and delivery vehicles, to achieve long-term autonomy. This need is highlighted by the increasing interest in developing mobile robots capable of coexisting and interacting safely and helpfully with humans. Anticipating likely motion trajectories allows autonomous agents to anticipate future movements of other agents and thus navigate safely as well as plan socially compliant motions by imitating observed trajectories.

Many methods predicting motion extrapolate partially observed trajectories, without incorporating knowledge of the environment. Example of such methods include constant acceleration [1], filtering methods [2], and auto-regressive models [3]. Advances in machine learning have lead to the development of methods which learn the motion behaviours from a dataset of trajectories. By training on motion data collected in the same environment, some learning-based methods can learn the general flow of movement [4]–[6], which implicitly accounts for environmental geometry. However, such methods are typically environment-specific, requiring predictions to be made in the same environment as that the training data was collected.

Through experience, humans have developed the ability to anticipate movement patterns based on the layout of the environment. We hypothesise that the structure of environments contains information about how objects move within the environment. By considering 2D environment floor plans, given as occupancy grid maps, it is possible to transfer motion patterns from training environments to new environments where no motion observations have been made. We propose a probabilistic generative model, Occupancy-Conditional Trajectory Network (OTNet), capable of generating motion trajectories for new unseen environments by generalising trajectories from previously observed environments, as shown in Figure 1. We empirically demonstrate that our model is capable of generating motion trajectories in simulated and real-world environments by motion behaviour observed in other environments.

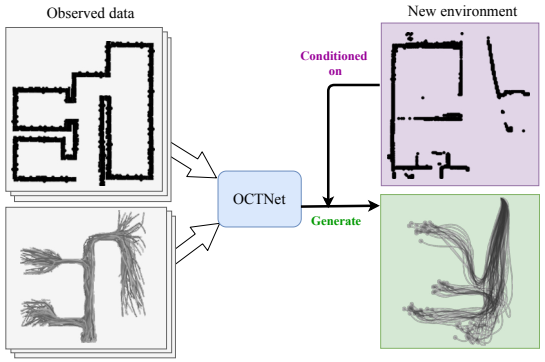

Fig. 1: We use OTNet to transfer motion patterns from maps where motion has been observed to new environments where motion has not been observed, by conditioning on the occupancy map of the new environment.

OTNet is a generative model which combines ideas from kernel methods and neural networks, with the following desirable properties:

1) It generalises motion patterns observed in previous environments, to generate motion trajectories in a new environment, where no motion has been observed;
2) It effectively models the probabilistic and multi-modal nature of motion generation. Individual trajectories can be generated from the model by sampling from it;
3) It generates individual trajectories as continuous functions, allowing trajectories to be queried at arbitrary resolution. We have the ability to specify fixed start-points for generated trajectories.

II. Related Work

A. Motion Trajectory Prediction

Estimating likely motion trajectories has been studied for a long time. Early simple methods to predict motion are often dynamics-based methods which extrapolate based on the laws

Correspondence to: W. Zhi, weiming.zhi@sydney.edu.au.
* Equal Contribution
[1] School of Computer Science, the University of Sydney, Australia
[2] Autonomous Systems Lab, ETH Zurich, Zurich, Switzerland
[3] NVIDIA, USA

of physics, such as constant velocity and constant acceleration models [1], with more complex Dynamics-based methods are utilised in [7], [8]. These models typically requires making a priori assumptions about agent motions, leaving little room to learn from observations. Other attempts at modelling motion trajectories include building dynamic occupancy grid maps based on occupancy data over time [9]–[11], inverse motion planning methods [12], and hidden Markov methods [13]. Recent developments in machine learning have led to an interest in data-driven models [14], which make fewer assumptions but rely heavily on observed data. A class of learning-based methods focus on learning a map of motion in a specific environment to implicitly capture map-aware motion patterns in the environment [4]–[6], [15].

B. Trajectory Generation

Neural network based generative models have been used in motion prediction, where they typically generate complete trajectories conditioned on an incomplete one. Generative adversarial networks (GANs) [16] are a popular class of generative models that for trajectory generation [14], [17]. Conditional Variational Autoencoders (CVAE) [18] are another class of generative models [19], [20] that have been utilised in the same problems. Similar to this work, ideas for transferring motion patterns in specific transport scenarios are considered in [21]. Methods using neural networks to generate trajectories based on observed trajectories have also gained attention in the motion planning community. A recent work, Motion Planning Network (MPNet) [22], aims to generate trajectories by learning from a dataset of optimal trajectories in various simulated environments.

III. Methodology

A. Problem Formulation

This paper addresses the problem of generating motion trajectories in new environments, by transferring trajectories data observed in other training environments. Figure 1 gives a high level idea of the problem at hand: We have observed data in the form of motion trajectories and maps for training (left), and provided an occupancy map of a novel environment (upper right), we wish to generate likely motion patterns (lower right).

We assume to have a training dataset consisting of occupancy grid maps of diverse environments and a collection of trajectories observed within each map. We denote the dataset as $\mathcal{D} = \{\mathcal{M}_n, \{\boldsymbol{\xi}_p\}_{p=1}^{P_n}\}_{n=1}^{N}$, where there are N maps and corresponding sets of observed trajectories, $\mathcal{M}_n$ is the n^{th} occupancy map, and $\{\boldsymbol{\xi}_p\}_{p=1}^{P_n}$ is the set of P_n trajectories collected in the corresponding environment. Note that the number of trajectories in each environment may differ.

We now have a new map, $\mathcal{M}^*$, where no trajectory observations have been made, and seek to generate new likely motion trajectories on $\mathcal{M}^*$, based on patterns we see in our training dataset. Key requirements of a solution to the problem include: (1) the ability to generate trajectories are diverse enough to capture multiple trajectory patterns; (2) the ability to generate trajectories that start from a specified coordinate.

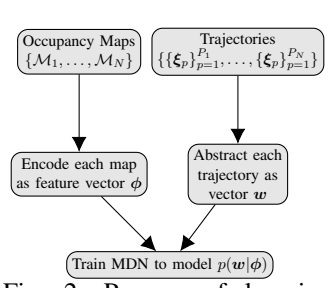

Fig. 2: Process of learning model to generate $p(\boldsymbol{w}|\boldsymbol{\phi})$

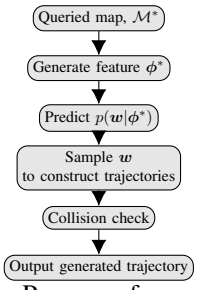

Fig. 3: Process of generating trajectories

B. Overview of OTNet

On a high level, our method encodes maps as similarities between themselves and a pre-defined collection of maps, and represents trajectories as parameters of a function. A mixture density network [23] is then used to predict distributions over function parameters, conditional on an encoded map.

The training process is illustrated in Figure 2, and can be summarised as:

1) Construct feature vectors of similarities, $\boldsymbol{\phi}$, from each training map. Intuitively, we similarly shaped floor-plans to have similar motion patterns. Our map encoding explicitly introduces the notion of map similarity to the learning process. Details in Section III-C.
2) Represent trajectory data as vectors of weight parameters, $\boldsymbol{w}$, of a function. This allows us to operate on on trajectory data sequences which contain differing numbers of waypoints. Each trajectory can typically be represented with much fewer parameters than waypoint coordinates. Details in Section III-D.
3) We use a mixture density network (MDN) [23] to learn the distribution over weight parameters conditioned on the map feature vectors, $p(\boldsymbol{w}|\boldsymbol{\phi})$. Details in Section III-E.

A brief overview of the generative process is illustrated in Figure 3.

After the MDN has been trained, we construct a feature vector $\boldsymbol{\phi}^*$ of a new map $\mathcal{M}^*$, and query the MDN to obtain $p(\boldsymbol{w}|\boldsymbol{\phi}^*)$. Vectors of $\boldsymbol{w}$ can be sampled from $p(\boldsymbol{w}|\boldsymbol{\phi}^*)$, and each sample $\boldsymbol{w}$ can be used to generate a new trajectory. As there are no explicit constraints in the MDN to prevent trajectories from overlapping with occupied regions, and we can efficient sample check for collisions, we accept collision-free trajectories to output. If we are only interested in trajectories that start at a certain point, we can generate trajectories conditional on a specified start-point.

C. Encoding of Environmental Occupancy

We represent occupancy maps as a vector of similarities between the given environment and a representative database of other maps. Intuitively, we can think of this as operating in the *space of maps*, where we pin-point the map of interest by its relation with other maps, where maps similar to one another are closer together in the space of map. We expect similar maps to have similar motion trajectory patterns. Related ideas

have been explored in the context of pseudo-inputs for sparse Gaussian process [24].

The Hausdorff distance is a widely used distance measure to shapes and images [25], and can be efficiently computed in linear time. The Hausdorff distance measures the distance between two finite sets of points, and allows us to make comparisons between our maps. We place the *Hausdorff distance* into a *distance substitute kernel* [26], to obtain our similarity function.

Given two sets of points $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$, and in general n and m are not required to be equal, the one-sided Hausdorff distance between the two sets is defined as:

$$\hat{\delta}_H(A, B) = \max_{a \in A} \min_{b \in B} ||a - b||. \tag{1}$$

The one-sided Hausdorff distance is not symmetric, we enforce symmetry by taking the average of $\hat{\delta}_H(A, B)$ and $\hat{\delta}_H(B, A)$, i.e.:

$$\delta_H(A, B) = \frac{1}{2}(\hat{\delta}_H(A, B) + \hat{\delta}_H(B, A)). \tag{2}$$

We can then define a similarity function between two sets A and B, analogous to a distance substitute kernel described in [26], as:

$$S_H(A, B) = \exp\Big\{ - \frac{\delta_H(A, B)^2}{2\ell_H} \Big\}, \tag{3}$$

where ℓ_H is a length scale hyper-parameter.

We extract occupied edge points of binary grid maps and evaluate the similarity function between each map in the representative database. We can select a set of maps within our training data to be in the representative database, or when the number of maps is low, we can consider all the training maps to be in the database. A simple method of selecting the set of examples to be in a representative database are described in [4]. If we have N training maps and M representative maps in the database, feature vector for the n^{th} map, $\boldsymbol{\phi}_n$, is:

$$\begin{bmatrix} \boldsymbol{\phi_1} \\ \vdots \\ \boldsymbol{\phi_N} \end{bmatrix} = \begin{bmatrix} S_H(\boldsymbol{\mathcal{M}}_1, \boldsymbol{\mathcal{M}_1}) & \ldots & S_H(\boldsymbol{\mathcal{M}}_1, \boldsymbol{\mathcal{M}}_M) \\ \vdots & \ddots & \vdots \\ S_H(\boldsymbol{\mathcal{M}}_N, \boldsymbol{\mathcal{M}}_1), & \ldots, & S_H(\boldsymbol{\mathcal{M}}_N, \boldsymbol{\mathcal{M}}_M). \end{bmatrix}. \tag{4}$$

For every map $\mathcal{M}$ in our dataset, there is a corresponding vector of similarities $\boldsymbol{\phi} \in \mathbb{R}^M$. The m^{th} element in the feature vector $\boldsymbol{\phi}_n$ denotes the similarity between the n^{th} occupancy map in the training dataset, and the m^{th} occupancy map in the database.

D. Continuous Representation of Trajectories

This section introduces the representation of trajectories in learning. Recorded trajectory data typically takes the form a sequence of discrete waypoints coordinates, whereas our generated trajectories are continuous functions. The continuous function representation allows for querying at arbitrary resolution without additional interpolation. We make the distinction between discrete and continuous trajectories:

1) **Discrete trajectories** are represented by an arbitrary-length sequence of waypoint coordinates. We denote a discrete trajectory, $\boldsymbol{\xi}$, with time steps $1 \ldots T$ as, $\boldsymbol{\xi} = \{(x_t, y_t)\}_{t=1}^T$, where (x_t, y_t) are x,y-coordinates of the dynamic object at time t.
2) **Continuous trajectories** are smooth continuous functions that map from $[0, 1]$ to coordinates. We define a continuous trajectory, $\boldsymbol{\Xi}$, as $\boldsymbol{\Xi}(\tau) = (x, y)$, where $\tau \in [0, 1]$, is a normalised time parameter.

We embed discrete trajectories, which are potentially of varying length, as fixed length vectors, allowing us to operate on trajectories of arbitrary length. The vectors correspond to weights of fixed basis functions which reconstructs a continuous trajectory that best fits the discrete trajectory.

We define a normalised timestep parameter $\tau \in [0, 1]$. A continuous trajectory can be modelled function, $\boldsymbol{\Xi}(\tau) = [x(\tau), y(\tau)]$, which maps τ to the x and y coordinates of the trajectory. We model $x(\tau)$ and $y(\tau)$ as weighted sums of fixed radial basis functions centred on evenly spaced τ values. Suppose we have a discrete trajectory $\boldsymbol{\xi} = \{(x_t, y_t)\}_{t=1}^T$, the weights that best fit a given discrete trajectory can be found by solving a pair of Kernel Ridge Regression (KRR) problems, defined as:

$$\arg\min_{\boldsymbol{w}_x} \sum_{t=1}^{T} (x_t - \boldsymbol{w}_x^T \boldsymbol{k}(\tau_t))^2 + \lambda ||\boldsymbol{w}_x||^2, \tag{5}$$

$$\arg\min_{\boldsymbol{w}_y} \sum_{t=1}^{T} (y_t - \boldsymbol{w}_y^T \boldsymbol{k}(\tau_t))^2 + \lambda ||\boldsymbol{w}_y||^2, \tag{6}$$

where $\tau_t = \frac{t}{T}$ is the normalised time parameter, λ is the ridge regularisation parameter, $\boldsymbol{k}(\tau_t)$ contains the radial basis function values evaluated at τ_t, obtained by:

$$\boldsymbol{k}(\tau) = k(\tau, \hat{\boldsymbol{\tau}}) = [k(\tau, \hat{\tau}_1), k(\tau, \hat{\tau}_2), \ldots, k(\tau, \hat{\tau}_M)]^T, \tag{7}$$

where $\hat{\boldsymbol{\tau}} = [\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_M]$ is a vector of τ values where the stationary radial basis functions are centred. In this work, we use the squared exponential basis function, as it is smooth and the default in many kernel based methods. Hence, our basis function is defined by

$$\boldsymbol{k}(\tau) = \Big[\frac{-||\tau - \hat{\tau}_1||^2}{2\ell_b}, \frac{-||\tau - \hat{\tau}_2||^2}{2\ell_b}, \ldots, \frac{-||\tau - \hat{\tau}_M||^2}{2\ell_b} \Big]^T, \tag{8}$$

where ℓ_b is the length scale hyper-parameter of the squared exponential function, and $|| \cdot ||$ denotes the L2-norm.

After evaluating Equation (8) to obtain $\boldsymbol{k}(\tau_t)$ for each τ_t considered, we can solve the KRR Equation (5), by computing:

$$\boldsymbol{w}_x = \Big(\lambda \boldsymbol{I} + \sum_{t=1}^{T} \boldsymbol{k}(\tau_t)^T \boldsymbol{k}(\tau_t) \Big)^{-1} \Big(\sum_{t=1}^{T} x_t \boldsymbol{k}(\tau_t) \Big), \tag{9}$$

$$\boldsymbol{w}_y = \Big(\lambda \boldsymbol{I} + \sum_{t=1}^{T} \boldsymbol{k}(\tau_t)^T \boldsymbol{k}(\tau_t) \Big)^{-1} \Big(\sum_{t=1}^{T} y_t \boldsymbol{k}(\tau_t) \Big), \tag{10}$$

where $\boldsymbol{I}$ is an identity matrix. We denote the concatenation of $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$ as $\boldsymbol{w} = [\boldsymbol{w}_x, \boldsymbol{w}_y]^T \in \mathbb{R}^{2M}$, where M is the number of basis functions.

Every discrete trajectory $\boldsymbol{\xi} = \{(x_t, y_t)\}_{t=1}^T$ can be converted to a corresponding vector of weight parameters $\boldsymbol{w} \in \mathbb{R}^{2M}$.

Typically there are significantly fewer weight parameters than waypoint coordinates required to represent a trajectory. We can then query the trajectory coordinates at τ^* by evaluating $x(\tau^*) = \boldsymbol{w}_x^T \boldsymbol{k}(\tau^*)$ and $y(\tau^*) = \boldsymbol{w}_y^T \boldsymbol{k}(\tau^*)$.

E. Learning a Mixture of Stochastic Processes

Recall that from Section III-C, each occupancy representation is encoded as a vector of similarities $\boldsymbol{\phi}$, and from Section III-D, all the trajectories observed in the map are embedded as a collection of fixed length weight vectors, $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_p\}$, where each $\boldsymbol{w}$ represents a trajectory. We aim to train a neural network to connect map representations and trajectory representations, and learn $p(\boldsymbol{w}|\boldsymbol{\phi})$.

There may exist many distinct groupings of trajectories in each environment. The distribution of trajectories is expected to be multi-modal, and we need to use a model capable of predicting multi-modal conditional distributions over the weights, $p(\boldsymbol{w}|\boldsymbol{\phi})$. Mixture density networks (MDN) [23] are a class of neural networks capable of representing conditional distributions. We slightly modify the classical MDN described in [23] to learn a mixture of vectors of conditional distributions, corresponding to the conditional distribution for each element in $\boldsymbol{w}$. We model the conditional distribution $p(\boldsymbol{w}|\boldsymbol{\phi})$ as a mixture of Q vectors of distributions, which we call mixture components. We make the mean-field Gaussian assumption [27] on the weights giving,

$$p(\boldsymbol{w}|\boldsymbol{\phi}) = \sum_{q=1}^{Q} \alpha_q p_q(\boldsymbol{w}|\boldsymbol{\phi}) = \sum_{q=1}^{Q} \alpha_q \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_q, \Sigma_q), \quad (11)$$

where $p_q(\boldsymbol{w}|\boldsymbol{\phi})$ denotes the q^{th} component, and α_q is the associated component weight. From the mean-field Gaussian assumption, each mixture component has mean vector, $\boldsymbol{\mu}_q = [\mu_{q,1}, \mu_{q,2}, \ldots, \mu_{q,2M}]^T$, and diagonal covariance, $\text{diag}(\Sigma_q) = \boldsymbol{\sigma}_q^2 = [\sigma_{q,1}^2, \sigma_{q,2}^2, \ldots, \sigma_{q,2M}^2]^T$. We can write each component of the conditional distribution as:

$$p_q(\boldsymbol{w}|\boldsymbol{\phi}) = \prod_{m=1}^{2M} \frac{1}{\sqrt{2\pi\sigma_{q,m}^2}} \exp\Big\{ -\frac{(w_m - \mu_{q,m})^2}{2\sigma_{q,m}^2} \Big\}, \quad (12)$$

giving us the negative log-likelihood loss function over N maps, and P_n trajectories observed in the environment corresponding to the n^{th} map in the dataset as:

$$\mathcal{L}(\boldsymbol{\theta}) = -\log \Big[\prod_{n=1}^{N} \prod_{p=1}^{P_n} \sum_{q=1}^{Q} \alpha_q p_q(\boldsymbol{w}|\boldsymbol{\phi}) \Big], \quad (13)$$

where we denote the set of parameters to optimise as $\boldsymbol{\theta} = \{\alpha_q, \boldsymbol{\mu}_q, \Sigma_q\}_{q=1}^{Q}$. Using a neural network to minimise the loss function defined in Equation (13), we can learn a model that maps from the feature vector of similarities $\boldsymbol{\phi}$ to the parameters required to construct $p(\boldsymbol{w}|\boldsymbol{\phi})$. The neural network is relatively simple with a sequence of fully-connected layers, the details of the architecture and parameters are shown in Figure 4.

The standard MDN constraints are applied using the activation functions highlighted in [23]. This includes:

1) $\sum_{q=1}^{Q} \alpha_q = 1$, such that component weights sum up to one, by applying the softmax activation function on associated network outputs;
2) $\sigma_{q,m} \geq 0$, by applying an exponential activation function on associated network outputs.

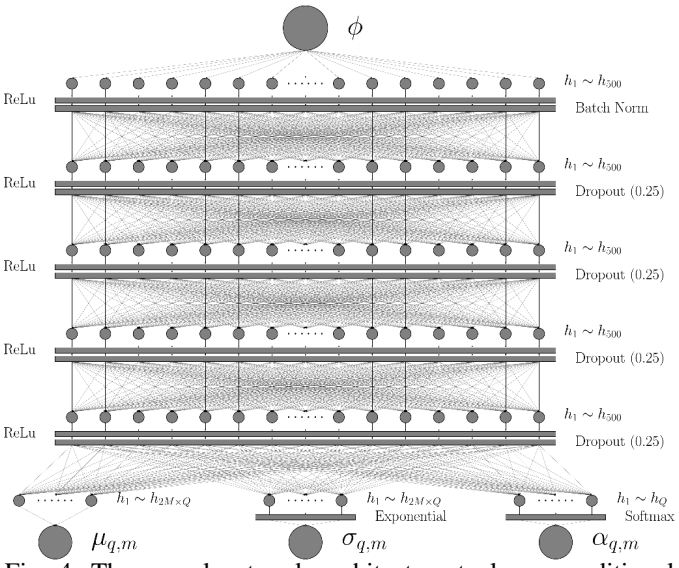

Fig. 4: The neural network architecture to learn conditional distributions over trajectory weight parameters $p(\boldsymbol{w}|\boldsymbol{\phi})$. The input is $\boldsymbol{\phi}$ and the outputs are $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\alpha}$. Hidden layers with n hidden units are denoted by h_1 h_n. ReLu, exponential, and softmax activation functions, along with, dropout, batch normalisation layers are also used.

As a distribution is estimated for each of the elements in weight vector, $\boldsymbol{w}$, the predicted $p(\boldsymbol{w}|\boldsymbol{\phi})$ results in a mixture of discrete processes, where each realisation is a vector of $\boldsymbol{w}$. $\boldsymbol{k}$ denotes the basis functions outlined in Section III-D.

F. Trajectory Generation and Conditioning

After we complete the training of our MDN model, we can generate trajectories in environments with no observed trajectories. We generate the feature vector of similarities, $\boldsymbol{\phi}^*$, from the map of interest, $\mathcal{M}^*$, and into the MDN. We obtain parameters, that define conditional distributions over $\boldsymbol{w}$. Realisations of $\boldsymbol{w}$ can be sampled randomly from the predicted $p(\boldsymbol{w}|\boldsymbol{\phi}^*)$, and a possible continuous trajectory, $\boldsymbol{\Xi}$, can be found by evaluating $\boldsymbol{\Xi}(\tau) = [x(\tau), y(\tau)] = [\boldsymbol{w}_x' \boldsymbol{k}(\tau), \boldsymbol{w}_y' \boldsymbol{k}(\tau)]$, where $\boldsymbol{k}(\tau)$ gives the basis function evaluations given in Section III-D, and $[\boldsymbol{w}_x', \boldsymbol{w}_y']^T$ is a realisation of $\boldsymbol{w}$. As there are no explicit constraints in the MDN to prevent the generation of trajectories which overlap with occupied regions, we apply collision checking. The trajectories can be generated and checked very efficiently, as it involves randomly sampling a mixture of Gaussian distributions and checking a map.

In order to predict how agents observed at a known position move, we are often interested in generating likely trajectories which begin at a certain start-point. To achieve this, let us consider the distribution of trajectories x, y-coordinates. Intuitively, rather than the distribution on weight parameters $\boldsymbol{w}$, we want to consider the joint distribution between trajectory coordinates. We evaluate the continuous trajectories at a set of

L times of interest, $\boldsymbol{\tau} = [\tau_1, \tau_2 \dots \tau_L]$,

$$p(\boldsymbol{\Xi}(\boldsymbol{\tau})) = p\Big(\begin{bmatrix} \boldsymbol{w}_x^T K(\boldsymbol{\tau}) \\ \boldsymbol{w}_y^T K(\boldsymbol{\tau}) \end{bmatrix} \Big) = \sum_{q=1}^{Q} \alpha_q \begin{bmatrix} \mathcal{N}(\hat{\boldsymbol{\mu}}_q^x, \hat{\Sigma}_q^x) \\ \mathcal{N}(\hat{\boldsymbol{\mu}}_q^y, \hat{\Sigma}_q^y) \end{bmatrix}, \quad (14)$$

where $K(\boldsymbol{\tau}) = [\boldsymbol{k}(\tau_1), \boldsymbol{k}(\tau_2), \dots, \boldsymbol{k}(\tau_L)]^T$ is a matrix containing basis function evaluations at L time points of interest $\boldsymbol{\tau}$. We can now find mean and covariance parameters $(\hat{\boldsymbol{\mu}}_q^x, \hat{\boldsymbol{\mu}}_q^y, \hat{\Sigma}_q^x, \hat{\Sigma}_q^y)$ for the trajectory coordinates at times of interest as,

$$\hat{\boldsymbol{\mu}}_q^x = \boldsymbol{\mu}_q^{xT} K(\boldsymbol{\tau}), \qquad \hat{\Sigma}_q^x = K(\boldsymbol{\tau}) \Sigma_q^x K(\boldsymbol{\tau})^T, \quad (15)$$

$$\hat{\boldsymbol{\mu}}_q^y = \boldsymbol{\mu}_q^{yT} K(\boldsymbol{\tau}), \qquad \hat{\Sigma}_q^y = K(\boldsymbol{\tau}) \Sigma_q^y K(\boldsymbol{\tau})^T, \quad (16)$$

The covariances of the q^{th} component between the elements in weight parameter vectors $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$ are denoted as Σ_q^x and Σ_q^y respectively. The component covariance Σ_q between all the weights $\boldsymbol{w} = [\boldsymbol{w}_x, \boldsymbol{w}_y]$ and Σ_q^x, Σ_q^y are $\mathrm{diag}(\Sigma_q) = [\mathrm{diag}(\Sigma_q^x), \mathrm{diag}(\Sigma_q^y)]$. Note that although Σ_q is diagonal, the covariance of Gaussian components between times of interest $\hat{\Sigma}_q^x, \hat{\Sigma}_q^y$ are typically dense.

For any component, we can then enforce locations that the trajectories must pass through (x^*, y^*) at τ_i, by finding the distributions of coordinates the times of interest conditioned on the fixed point. Consider when we wish to condition on the start point, i.e. $\tau_1 = (x^*, y^*)$, a single component in the mixture can be written as:

$$p_q(\boldsymbol{\Xi}(\boldsymbol{\tau}_{1:L}) | \boldsymbol{\Xi}(\boldsymbol{\tau}_1) = [x^*, y^*]) = \alpha_q \begin{bmatrix} \mathcal{N}(\bar{\boldsymbol{\mu}}^x, \bar{\Sigma}^x) \\ \mathcal{N}(\bar{\boldsymbol{\mu}}^y, \bar{\Sigma}^y) \end{bmatrix}. \quad (17)$$

If we consider elements in mean vectors $\hat{\boldsymbol{\mu}}^x$, $\hat{\boldsymbol{\mu}}^y$ and covariance matrices $\hat{\Sigma^x}$, $\hat{\Sigma^y}$ of the unconditioned distribution as:

$$\hat{\boldsymbol{\mu}}^z = \begin{bmatrix} \hat{\boldsymbol{\mu}}_1^z \\ \hat{\boldsymbol{\mu}}_{2:L}^z \end{bmatrix}, \quad \hat{\Sigma}^z = \begin{bmatrix} \hat{\Sigma}_{1,1}^z & \hat{\Sigma}_{1,2:L}^{z\,T} \\ \hat{\Sigma}_{1,2:L}^z & \hat{\Sigma}_{2:L,2:L}^z \end{bmatrix}, \quad \text{for } z = \{x, y\}, \quad (18)$$

then for $z = \{x, y\}$, we can express the parameters of the conditional distribution as [27]:

$$\bar{\boldsymbol{\mu}}^{\boldsymbol{z}} = \hat{\boldsymbol{\mu}}_{2:L}^z + \hat{\Sigma}_{1,2:L}^z \hat{\Sigma}_{1,1}^{z\,-1} (z^* - \hat{\boldsymbol{\mu}}_1) \quad (19)$$

$$\bar{\Sigma}^z = \hat{\Sigma}_{2:L,2:L}^z - \hat{\Sigma}_{1,2:L}^z \hat{\Sigma}_{1,1}^{z\,-1} \hat{\Sigma}_{1,2:L}^{z\,T}. \quad (20)$$

We can condition each mixture component to start at a certain point, or employ strategies to only condition selected mixture components, such as the nearest component to the conditioned coordinate, and only generate trajectories belonging to the conditioned mixture component.

IV. Experimental Results and Discussion

A. Dataset and Metrics

Training an OTNet requires a dataset containing occupancy maps of multiple environments along with observed trajectories in each environment. To the best of our knowledge, there exists no real-world dataset of sufficient size with different occupancy maps and trajectories observed in each of the different environments. Therefore, we conduct our experiments with our simulated dataset, *Occ-Traj120* [28]. This dataset contains 120 binary occupancy grid maps of indoor environments with rooms and corridors, as well as simulated motion trajectories.

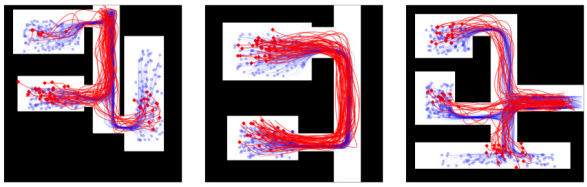

Fig. 6: Generated likely motion trajectories (in red) in new environments. End points are indicated by scatter points. OTNet captures the probabilistic, multi-modal nature of motion trajectories. The ground truth trajectories (in blue) are hidden during training.

We aim to utilise our method to learn transfer the motion to new environments. The dataset is split with a 80-20 ratio for training and testing respectively. Our aim is to transfer the trajectory behaviour, from a training subset to unseen test environments.

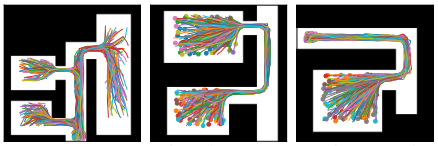

Fig. 5: Occupancy maps in the Occ-Traj 120 dataset, along with associated trajectories. We aim to generalise trajectories to unseen maps.

We evaluate the generated trajectories against a test set with hidden ground truth trajectories. Continuous trajectories outputted are discretised for evaluation by querying at uniform intervals. Due to the probabilistic and multi-modal nature of our output, the metric used is *minimum trajectory distance* (MTD), and is defined by: $MTD = \min_{i=1,\dots,P} \mathcal{D}(\xi_{gen}, \xi_i)$, where P denotes the number of trajectories observed in the environment, with i indexing each trajectory, and $\mathcal{D}(\xi_{gen}, \xi_i)$ is a distance measure of trajectory distance between the generated ξ_{gen} and a ground truth trajectory ξ_i. In our evaluations the Euclidean *Hausdoff distance* and *discrete Frechét distance* are considered. These trajectory distances are commonly used in distance-based trajectory clustering to quantify the dissimilarity between trajectories, and a review of these distances can be found in [29]. We also wish to evaluate the quality of the uncertainty captured. To this end, we calculate the average negative log-likelihood (ANLL) between the predicted distribution at 100 uniform time-steps and each ground truth trajectory. When only samples of generated trajectories are available, we fit Gaussian distributions over world-space coordinates at uniform time-coordinates. ANLL can take into account of probabilistic multi-modal distributions, and a relative lower ANLL indicates better performance.

B. Experimental Setup

We compare our proposed method, OTNet, with neural-network based generative models, a learning-based motion reconstruction method, and a k-nearest neighbour (k-NN) based method. The details of these models are as follows:

1) OTNet: We train OTNet with the length-scale hyper-parameters $\ell_H = 50$ and $\ell_b = 5$ for 20 epochs. The number of bases are set to be $M = 15$, and the number of mixture components, $Q = 4$. These hyper-parameters are hand-picked and fairly not sensitive, with values in the same ball-park giving similar results. Cross-validation could be applied for further improvements. As the number of map examples in the training set is relatively small at 96, we select all our training maps to be in our database of maps, when encoding maps of interest as feature vectors. In each of our experiments, trajectories can be generated efficient during test time. On a standard desktop machine, predicting the distribution of trajectories can be done in under one second, negligible time is required to randomly sample the distribution to generate trajectories.

2) Generative network models: Our proposed method is generative, we evaluate two popular generative models: GANs [16] CVAEs [18], trained for 300 epochs to generate trajectory parameters $\boldsymbol{w}$. The discriminator uses convolutional layers, and the generator samples a 100-dimensional latent $\hat{\mathbf{z}}$, concatenated with the map for conditioning. Five dense layers are used to output $\boldsymbol{w}$. The hyperparameters of the CVAE model, are identical to the GAN model.

3) Learning-based motion reconstruction: We evaluate the performance of the learning-based MPNet [22], which conditions on the environment and imitates training trajectories. We pre-train on MPNet's public 2D dataset with further training with the *Occ-Traj120 dataset* for 300 epochs. MPNet requires trajectory start and end coordinates, during evaluation we provide ground truth mean start and end points of groups of trajectories. To guarantee valid trajectories, we use the hybrid MPNet-RRT variant suggested by the authors. We emphasise that MPNet can only generate a single prediction, and cannot generate distributions of trajectories.

4) Nearest-neighbour: We also consider a baseline k-NN based approach. Using the symmetric Hausdoff distance, we find the nearest map training map, and transfer the trajectories directly to the queried map. As we cannot average trajectories, we use $k = 1$. Note storage of not only all maps, but also trajectories are required.

C. Comparisons to Other Generative Models

	Hausdoff	Frechet	ANLL
OTNet	1.98	2.13	29.83
GANs	11.79	16.66	NA
CVAE	9.48	14.67	NA
MPNet-RRT	2.99	5.14	NA
k-NN	2.03	2.14	31.46

TABLE I: Performance of OTNet and comparisons models. Lower values are better. OTNet outperforms the comparisons, although the performance of k-NNs are strong, k-NNs lack the ability to generate new trajectories that do not exist in the provided dataset.

The performance results of our experiments are tabulated in Table I, we see that OTNet outperform the other generative models compared. The ANLL for CVAE and GANs suffer from a loss of precision due to an extremely low likelihood, resulting in taking log of a near 0 value, the ANLL will be much higher than OTNet. The ANLL for MPNet-RRT is also not applicable, as it generates single trajectories. The map representations are of too high dimensions to directly train neural networks with a limited dataset. In particular, the encoding of each occupancy map as a feature vector of similarities, $\boldsymbol{\phi}$, allows for flexible representations even when the number of maps in the dataset is small. Comparatively, other neural network-based models which attempt to directly extract map features using convolutional neural networks are unable to successfully condition on the occupancy maps. The nearest neighbour model, which transfers the trajectories of the nearest map, performs surprising well, comparable to the OTNet on the simulated on the Haussdoff and Frechét measures. This is due to OTNet often containing several relatively similar-looking maps. However, OTNet is able to take into consideration several maps, and weigh each by their similarity with the map of interest, while the nearest neighbour only considers the closest map. We also note that the single nearest neighbour approach often results in relative over-confidence of biased predictions, hence the higher ANLL relative to OTNet. We note that we can directly obtain closed form equations of trajectory distributions from OTNet, allowing us to repeatedly generate new trajectories. Whereas, MPNet-RRT produces single predictions. Although using k-NN can give trajectories with competitive performance, it cannot generate new trajectories, whereas OTNet is capable of generating trajectories that have not been observed in the original dataset. Examples of trajectories generated by OTNet are shown in red in Figure 6, along with ground truth trajectories (blue).

D. Diverse Trajectories and Conditioning Trajectory Start Points

We also want to further investigate whether generated trajectories capture different groups of motion, and whether we can specify start-points for generated trajectories. Figure 8 shows generated trajectories in an unseen test environment, along with plots of the trajectory coordinates wrt the normalised time parameter, τ. The test environment is an indoor environment of a corridor with a connected room. Predicted trajectories are coloured in red, and hidden ground truth trajectories are underlaid in blue. The end coordinates of each trajectory have an attached marker. We clearly observe that the trajectories generated can belong to different groupings – one group starts from inside the room and exit into the corridor, while the other starts in the corridor and end in the room. The multi-modality of the distributions of trajectories learned is even more clear when observing the plots of coordinates against τ. If we observe the y-coordinates, $y(\tau)$, we clearly see two groupings of functions, one of which is at around $x(0) = 15$ and around $y(1) = 32$, the other grouping has values around $x(0) = 32$ and $y(1) = 15$.

In the same environment, we attempt to condition the generated trajectories to different start-points. The generated trajectories (top), as well as trajectory coordinates $x(\tau)$ (middle) and $y(\tau)$ (bottom) of the trajectories, are shown in Figure 9.

Fig. 7: After training on a simulated dataset, we condition on sections of real world occupancy data, from the Intel-Labs dataset [30], to generate motion trajectories (top two rows, green). The start point of the trajectories are fixed, with the end-points illustrated with magenta markers. We can compare this to trajectories from a baseline nearest neighbour model (bottom row, blue), which transfers trajectories from the most similar training map, and is unable to adapt to the new environments.

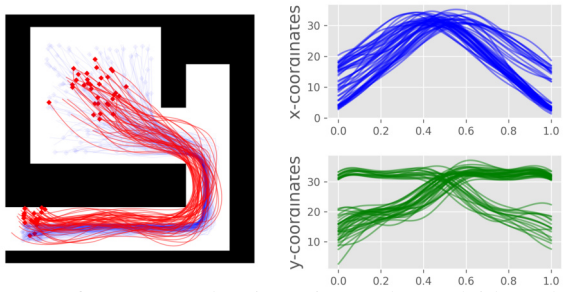

Fig. 8: (Left) Generated trajectories (red) overlaid on the corridor and room test map, with markers indicating the end-points. The hidden ground truth trajectories are under-laid in blue. We see two different groups of trajectories: one starts from inside the room and end in the corridor, the other moves in the opposite direction. (Right) Plots of coordinates wrt τ, $x(\tau)$ and $y(\tau)$ (top, bottom respectively), corresponding to the trajectories generated on the left plot. The ability to generate distinct groups of trajectories is clear.

The generated trajectories are in red, with red marker end-points, and black start-points. In the left and centre plots, we condition the trajectories' start point on two coordinates in the room, $(x(0) = 5, y(0) = 20)$ (left) and $(x(0) = 5, y(0) = 5)$ (centre), as well conditioning on a start-point in the corridor, $(x(0) = 2, y(0) = 35)$. We see that in all three cases, OTNet was able to generate reasonable trajectories, which follow the environment structure, with each starting at their designated start coordinates.

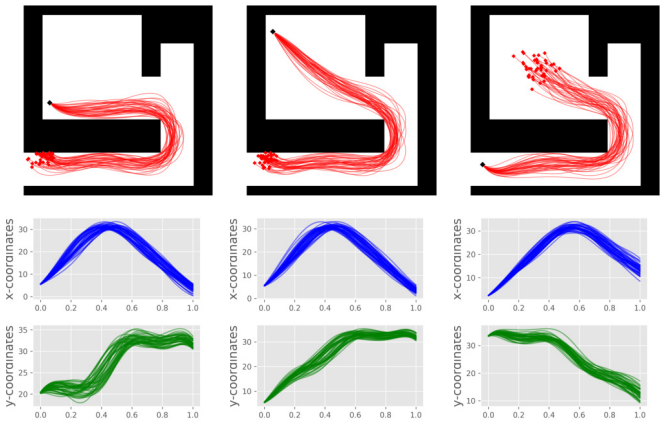

Fig. 9: We generate trajectories (in red, with red markers as endpoints) conditional on the start points (black marker). We condition on two start points inside the room (left, centre), and one point in the corridor (right). The corresponding plots of x, y-coordinates wrt to τ, $x(\tau)$ and $y(\tau)$ (top, bottom respectively). We see that the trajectories can be conditioned on a variety of start points.

E. Transferring Trajectories in Simulated Environments to Real-world Data

We also investigate the transfer of trajectories from simulated maps to real-world environments. We select three different regions in the Intel-lab dataset [30], and transfer trajectory patterns trained on the simulated Occ-Traj dataset. Figure 7 shows trajectories (top two rows, green with magenta end-points) generated on three different sub-maps, and conditioned on two different starting locations. We see that OTNet is able to generate multi-modal distributions of trajecto-

ries, with distinct groupings, and largely conforms to the environmental geometry, even though the real-world occupancy differs from the simulated training environments significantly. We visually compare trajectories generated by OTNet to those by the 1-nearest neighbour approach (bottom row, blue with magenta end-points), which transfers the trajectories from the most similar map to the queried map. Qualitatively, we evaluate the trajectories generated by OTNet to conform better to the real-world environment compared to the nearest-neighbour trajectories. The 1-nearest neighbour approach can only draw from the most similar map. We note that the 1-nearest neighbour approach performs well when transferring to the simulated dataset, as it could often find a relatively similar map that has been observed. Whereas, the real-world occupancy map is not very similar to any single map we have experienced, but OTNet can generalise trajectories from a combination of maps which resembles the test environment closer. Hence, OTNet is more capable of generalising to unseen environments. The nearest neighbour approach is also unable to immediately generate similar new trajectories beyond those included in the data, nor is it possible to condition predicted trajectories on a start-point. Both generating similar new trajectories, and conditioning on points can be achieved with OTNet.

V. Conclusion

We present a novel generative model, OTNet, capable of producing likely motion trajectories in new environments where no motion has been observed. We generalise observed motion trajectories in alternative training environments. The OTNet encodes maps as a feature vector of similarities, and embeds observed trajectories as function parameters. A neural network is used to learn conditional distributions over the parameters. Realisations of the vectors can then be sampled from the conditional distribution, and used to reconstruct generated trajectories. We empirically show the strong performance of OTNet against popular generative methods. Future improvements on OTNet include incorporating temporal changes in trajectory patterns into the framework.

References

[1] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, 2003.

[2] Y. Zhang, Z. Zhai, X. Nie, C. Ma, and F. Zuo, "An extended self-adaptive kalman filtering object motion prediction model," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2008.

[3] A. Agarwal and B. Triggs, "Tracking articulated motion using a mixture of autoregressive models," in *ECCV*, 2004.

[4] W. Zhi, R. Senanayake, L. Ott, and F. Ramos, "Spatiotemporal learning of directional uncertainty in urban environments with kernel recurrent mixture density networks," *IEEE Robotics and Automation Letters*, 2019.

[5] T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal, "Enabling flow awareness for mobile robots in partially observable environments," *IEEE Robotics and Automation Letters*, 2017.

[6] R. Senanayake and F. Ramos, "Directional grid maps: modeling multimodal angular uncertainty in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[7] S. Zernetsch, S. Kohnen, M. Goldhammer, K. Doll, and B. Sick, "Trajectory prediction of cyclists using a physical model and an artificial neural network," in *IEEE Intelligent Vehicles Symposium (IV)*, 2016.

[8] J. F. Kooij, F. Flohr, E. A. Pool, and D. M. Gavrila, "Context-based path prediction for targets with switching dynamics," *Int. J. Comput. Vision*, 2019.

[9] D. Arbuckle, A. Howard, and M. Mataric, "Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[10] N. C. Mitsou and C. Tzafestas, "Temporal occupancy grid for mobile robot dynamic environment mapping," 2007.

[11] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation," in *IEEE International Conference on Robotics and Automation*, 2014.

[12] B. D. Ziebart, N. D. Ratliff, G. Gallagher, C. Mertz, K. M. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. S. Srinivasa, "Planning-based prediction for pedestrians," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[13] D. Vasquez, T. Fraichard, and C. Laugier, "Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion," *The International Journal of Robotics Research*, 2009.

[14] A. Gupta, J. E. Johnson, F. Li, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] S. M. Mellado, G. Cielniak, T. Krajník, and T. Duckett, "Modelling and predicting rhythmic flow patterns in dynamic environments," in *TAROS*, 2018.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[17] J. Li, H. Ma, and M. Tomizuka, "Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning," *International Conference on Robotics and Automation (ICRA)*, 2019.

[18] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, pp. 3483–3491, 2015.

[19] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, "Generative modeling of multimodal multi-human behavior," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[20] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[21] N. Jaipuria, G. Habibi, and J. P. How, "A transferable pedestrian motion prediction model for intersections with different geometries," *ArXiv*, 2018.

[22] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[23] C. M. Bishop, "Mixture density networks," tech. rep., Aston University, 1994.

[24] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, 2006.

[25] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

[26] B. Haasdonk and C. Bahlmann, "Learning with distance substitution kernels," in *DAGM-Symposium, Lecture Notes in Computer Science*, 2004.

[27] C. M. Bishop, *Pattern Recognition and Machine Learning*. 2006.

[28] T. Lai, W. Zhi, and F. Ramos, "Occ-traj120: Occupancy maps with associated trajectories," *CoRR*, 2019.

[29] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[30] D. Hähnel, W. Burgard, D. Fox, and S. Thrun, "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.