The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Non-Parametric Learning for Natural Plan Generation

Ian Baldwin and Paul Newman
Oxford University Mobile Robotics Group
Email: {ib,pnewman}@robots.ox.ac.uk

***Abstract*— We present a novel way to learn sampling distributions for sampling-based motion planners by making use of expert data. We learn an estimate (in a non-parametric setting) of sample densities around semantic regions of interest, and incorporate these learned distributions into a sampling-based planner to produce natural plans. Our motivation is that certain aspects of the workspace have a local influence on planning strategies, which is dependent both on where, and *what*, they are. In the event that learning the density estimate of the training data is impractical in the original feature space, we utilize a non-linear dimensionality-reduction technique and perform density estimation on a lower-dimensional embedding. Samples are then lifted from this embedded density into the original feature space, producing samples that still well approximate the original distribution.**

A goal of this work is to learn how various features in the environment influence the behavior of experts - for example, how pedestrian crossings, traffic signals and so on affect drivers. We show that learning sampling distributions from expert trajectory data around these semantic regions leads to more natural paths that are measurably closer to those of an expert. We demonstrate the feasibility of the technique in various scenarios for a virtual car-like robotic vehicle and a simple manipulator, contrasting the differences in planned trajectories of the semantically-biased distributions with conventional techniques.

I. Introduction

Sampling-based motion planning is an established technique in robotics that enables planning in complex workspaces, or for systems that have high-dimensional configuration spaces ($\mathcal{C}$-spaces). **R**apidly-exploring **R**andom **T**rees (**RRT**'s [1]) and **P**robabilistic **R**oad **M**aps (**PRM**'s [2]) are both instantiations of stochastic planners, both of which utilise a graphical representation of the workspace which captures the relationship between free and occluded space.

RRTs have been effective for a number of different motion planning tasks, including manipulation and grasping [3], kinodynamic planning[4] and for other non-holonomic systems [5]. However, uniform sampling of the configuration space suffers from the so-called "narrow passage problem" - more samples are generated in the free space than in the more complex regions, which is typically where more information is required.

Some of the earliest strategies for biasing sampling distributions include the Narrow Passage Sampling strategy [6] and the Gaussian Strategy for PRM's [7], amongst others. Both methods utilise a pre-determined criteria to determine whether samples generated in the $\mathcal{C}$-space or workspace are considered for planning. $\mathcal{C}$-space sampling strategies include Approximated Medial Axis sampling [8](which has a workspace counterpart), in addition to strategies which explore the $\mathcal{C}$-space boundaries[9]. Recent techniques have utilized features in the environment to better determine how to bias the sampling distribution - Adaptive Workspace Biasing [10] generates an optimized weighting of feature vectors defined over a voxelised workspace.

In this work, we consider the robot operator to be an expert, and seek to adapt the behavior of the planning algorithm so that the resultant solution paths are similar to those generated by the operator, whilst still maintaining the exploratory nature of the probabilistic approach. We show that:

1) it is possible to learn representations of sampling distributions and use these representations within a probabilistic planner to generate more natural plans
2) we are able to produce density estimates even when the training data have a distinct manifold structure
3) the plans obtained using a semantically-biased approach are measurably closer to those of an expert

II. Vehicle Planning

Using RRTs for vehicle-planning has been explored by numerous authors [11][12]. RRTs are appealing as their computational cost and complexity, for vehicles in $\mathbb{SE}(2)$, is relatively low. However, the paths generated by the RRT are only constrained by the collision-checker - if a given configuration is returned as collision-free, the algorithm assumes that it is a valid node. This may be undesirable (consider a vehicle being oriented perpendicular to oncoming traffic in an intersection), and we therefore seek a better informed sampling distribution around pertinent features.

We assume that the trajectories generated by a human operator incorporate an underlying probability distribution that is implicitly better to sample from regularly. As a motivating example, consider Figure 1 which shows a common scenario in the driving domain [1]- a traffic circle.

Trajectory data $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ was recorded from the manual operation of a virtual vehicle around the traffic circle, depicted in Figure 1, where each trajectory $\mathcal{D}_i$ consists of positional, directional, and speed data:

$$\mathcal{D}_i = \{\mathbf{x}, \mathbf{y}, \boldsymbol{\Theta}, |\mathbf{v}|\} \quad (1)$$

where $\mathbf{x} = \{x_1, \ldots, x_{|\mathcal{D}_i|}\}$, and similarly for $\mathbf{y}, \boldsymbol{\Theta}$ and $|\mathbf{v}|$. The overhead view of a sub-sampled set of this data is depicted in Figure 2(a), which also shows the $\langle \mathbf{x}, \mathbf{y}, \boldsymbol{\Theta} \rangle$ components of the data (b).

It is apparent from the data that the presence of the traffic circle influences driving behavior around it, and we seek to learn about this influence. Ideally we would like to generate samples for our probabilistic planner by sampling from:

[1]Note: This example assumes a left-handed driving convention

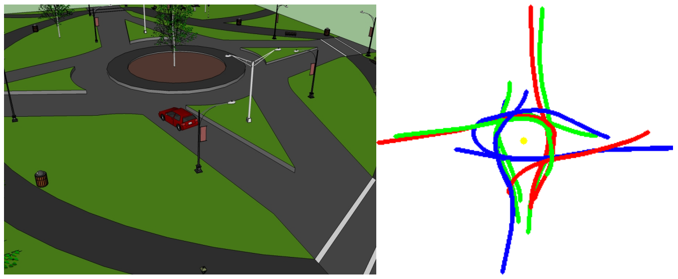
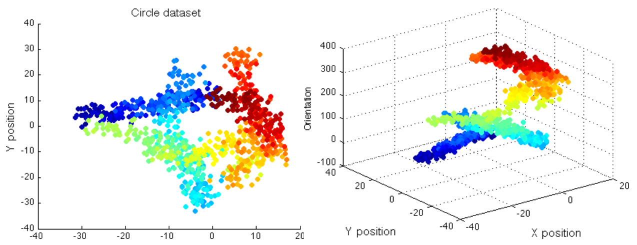

(a) (b)

Fig. 1: A typically encountered scenario in the driving domain - a traffic-circle (a). Also depicted is a set of trajectories around the circle obtained from recording data of a virtual vehicle under expert control (b).



(a) (b)

Fig. 2: The overhead view of a sample dataset, (a) drawn from the set of training data, $\mathcal{D}$, around the traffic-circle shown in Figure 1. $[0,0]$ represents the traffic circle centre. (b) represents a 3D view of the same data

$$p(\mathbf{x}, \mathbf{y}, \mathbf{\Theta}, |\mathbf{v}| \mid \mathcal{D}) \tag{2}$$

However, as can be seen from Figure 2, the distribution we seek to learn is complex. Estimating the density with a conventional technique such as the Gaussian Mixture Model (GMM) requires estimating the number of mixtures, k, and performing model-averaging to compare the fits for different values of k. Using a non-parametric density estimation technique, the Infinite Gaussian Mixture Model (IGMM) allows us to learn the value for k in a Bayesian setting.

By learning this Mixture Model, we can generate samples during planning that are more representative of what we have seen during training, leading to solutions more like those of the "expert". The advantage of employing this method is that learning is done off-line - once we have a set of n Gaussians $\mathcal{G} \sim \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ with appropriate weights $\pi = \{\pi_1, \ldots, \pi_n\}; \sum_{i=1}^{n} \pi_i = 1$ which approximate our distribution, generating N representative samples is a simple procedure described in Algorithm 1.

We now turn to the inference of the weightings and parameters of the Gaussians in the mixing model.

III. Learning the Sampling Distribution

There are numerous approaches to non-parametric density estimation, including Parzen windows, Dirichlet Diffusion Trees[13] and the Gaussian Process Density Sampler (GPDS)[14]. More conventional methods such as Parzen windows require estimation of a fundamental parameter of the distribution known as the *bandwidth*, whilst other

Algorithm 1 Sampling from a mixture model

```
1: procedure SampleFromMM($N$)
2:     $\pi_s = buildCumulativeDensity(\pi)$
3:     for $i = 1 : N$ do
4:         $r \leftarrow Random(0,1)$
5:         $i \leftarrow \arg\min_{j=1}^{|\pi|} \left(\sum_{i=1}^{j} \pi_{s(i)} \geq r\right)$
6:         $s_i \leftarrow sampleGaussian(\mathcal{G}_i)$
7:     end for
8: end procedure
```

estimation methods (such as the GPDS) are intended for estimation of very high-dimensional data. Non-parametric density estimation methods based on the Dirichlet Process allow for models of infinite complexity, whilst still being Bayesian and avoiding over-fitting - one example of which is the IGMM. We use the IGMM as implemented by Rasmussen [15] in order to learn the joint density. The IGMM allows for modeling highly-complex distributions, whilst still offering certain analytical advantages by utilising Gaussian distributions. A conventional Gaussian mixture model is defined as:

$$p(\mathcal{D} \mid \mu_1, \ldots, \mu_k, s_1, \ldots, s_k, \pi_1, \ldots, \pi_k) = \sum_{i=1}^{k} \pi_i \mathcal{N}(\mu_i, s_i^{-1}) \tag{3}$$

where k is the number of mixture components in the model, $\mu_1, \ldots, \mu_k$ represent the means of the Gaussian distributions used in the mixture, $s_1, \ldots, s_k$ represent the precisions, and $\pi_1, \ldots, \pi_k$ are the mixing coefficients. This model requires k to be known *a-priori* (or estimated from the data via Maximum Likelihood or Expectation Maximization), and this can limit the flexibility of the approach. The goal of the infinite mixture model is to place a prior allowing infinitely many mixtures over the classes, and then infer the correct number from the observed data. The assumption under the IGMM is that the observed data $\mathcal{D}$ (the trajectory data in Figure 2) are generated by the following process:

$$D_{i,j} \mid c_j \sim \mathcal{N}(\mu_{c_j}, s_{c_j}^{-1}) \tag{4}$$

where $\mathcal{D}_{i,j}$ indexes into the training data to give $\langle x, y, \theta, |v| \rangle$ data at position j in trajectory i. c_j is an indicator variable denoting membership of tuple $\mathcal{D}_{i,j}$ to mixing component j. Taking a Bayesian approach to the problem, the IMM places priors over the various elements of the Mixture Model and infers the number of mixture components (along with their associated means and precisions). The joint conditional distribution we seek to learn is:

$$p(\mu, \mathbf{s}, \pi \mid \mathcal{D}) \propto p(\mathcal{D} \mid \mu, \mathbf{s}, \pi) p(\mu, \mathbf{s}, \pi) \tag{5}$$

where μ are the component means, $\mathbf{s}$ are the precisions and π are the mixing components. The means are given a multivariate Gaussian prior:

$$\mu_i \sim \mathcal{N}(\lambda, r^{-1}) \tag{6}$$

The hyperparameters λ and r^{-1} are common to all components, and are initialised with vague Normal and

Gamma priors:

$$p(\lambda) \sim \mathcal{N}(\mu_y, \sigma_{\mathcal{D}}^2) \quad (7)$$
$$p(r) \sim \mathcal{G}(1, \sigma_{\mathcal{D}}^{-2}) \quad (8)$$

where $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}^2$ are the mean and variance of the trajectory data we have observed. The conditional distribution of the hyperparameters (taking Equation 3 as the likelihood term) can be shown to be:

$$p(\lambda \mid \mu, r) \sim \mathcal{N}\left(\frac{\mu_{\mathcal{D}}\sigma_{\mathcal{D}}^{-2} + r\sum_{i=1}^{k}\mu_i}{\sigma_{\mathcal{D}}^{-2} + kr}, \frac{1}{\sigma_{\mathcal{D}}^{-2} + kr}\right) \quad (9)$$

$$p(r \mid \mu, \lambda) \sim Wishart\left(b, \left[b^{-1}(\sigma_{\mathcal{D}}^2 + \sum_{i=1}^{k}(\mu_i - \lambda)^2\right]^{-1}\right) \quad (10)$$

The precisions are given Wishart priors:

$$p(s_i \mid \beta, w) \sim Wishart(\beta, w^{-1}) \quad (11)$$

where again β and w are common hyperparameters, described by the following Gamma and Wishart distributions:

$$p((\beta - \mathbf{d} + 1)^{-1}) \sim \mathcal{G}\left(\frac{1}{d}, 1\right) \quad (12)$$
$$p(w) \sim Wishart(1, \sigma_{\mathcal{D}}^2) \quad (13)$$

Where $\mathbf{d}$ is the dimensionality of the data, which in the case of the training data in Figure 2 is 4. Using Equation 11 as the likelihood term, the posterior conditional distribution for w, using the conjugate properties of the Wishart, are again distributed Wishart. Unfortunately, the posterior conditional for β is not a standard distribution, however Rasmussen shows that as the distribution $p(log(\beta) \mid s_1, \ldots, s_k, w)$ is log-concave, samples can be generated for $p(log(\beta))$ using Adaptive Rejective Sampling and then inverted to obtain samples for β. The conditional posterior precisions are given by:

$$\begin{aligned} p(s_i \mid \mathbf{c}, \mathcal{D}, \mu_i, \beta, w) &\sim Wishart(\phi, \psi) \quad \textbf{where} \quad (14) \\ \phi &= \beta + n_j \\ \psi &= \frac{1}{\phi}(w\beta + \sum_i (\mathcal{D}_i - \mu_j)^2)^{-1} \end{aligned}$$

For a conventional mixture model, the mixing coefficients are assigned a Dirichlet prior:

$$p(\pi_1, \ldots, \pi_k \mid \alpha) \sim Dirichlet\left(\frac{\alpha}{k}, \ldots, \frac{\alpha}{k}\right) \quad (15)$$

with a "concentration" parameter $\frac{\alpha}{k}$. In the infinite case, only a certain proportion of components will have training data associated with them, and these are termed the "represented" components. In order to avoid working with an infinite number of mixing weights, the joint conditional over the latent indicator variable c is introduced, defined by:

$$p(c_1, \ldots, c_N \mid \pi_1, \ldots, \pi_k) = \prod_{i=1}^{k} \pi_i^{n_i} \quad (16)$$

where n_i is the number of tuples associated with component i. It is then possible to integrate out the mixing weights, so that the indicators are only dependent on α:

$$p(c_1, \ldots, c_n \mid \alpha) = \frac{\Gamma(\alpha)}{\Gamma(N + \alpha)} \prod_{i=1}^{k} \frac{\Gamma(n_i + \frac{\alpha}{k})}{\Gamma\left(\frac{\alpha}{k}\right)} \quad (17)$$

where $N = |\mathcal{D}_{i=1:|\mathcal{D}|}|$ is the total number of training tuples. This allows for the inference of finitely many indicator variables as opposed to an infinite number of mixing weights. Because exact inference of the posterior distributions is infeasible, Gibbs sampling will be used to generate samples. We have already generated the conditional distributions for $\lambda, r, \mathbf{s}, w, \beta$, and therefore it only remains to obtain the conditional posterior of each indicator in turn, keeping all other indicators fixed:

$$p(c_i = j \mid \mathbf{c}_{-i}, \alpha) = \frac{n_{-i,j} + \frac{\alpha}{k}}{n - 1 + \alpha} \quad (18)$$

where $\mathbf{c}_{-i}$ indicates all indicator variables except i, and $n_{-i,j}$ is the number of tuples that are associated with component j apart from tuple i. As $k \to \infty$, the conditional prior over c_n will produce:

$$p(c_n = j \mid \mathbf{c}_{-n}, \alpha) = \frac{\alpha}{n - 1 + \alpha} \quad (19)$$

for the *represented* components (those components with data associated) and:

$$p(c_n = j \mid \mathbf{c}_{-n}, \alpha) = \frac{n_{-n,j}}{n - 1 + \alpha} \quad (20)$$

for the unrepresented components where c_n is the indicator for mixture n. The entire Gibbs sampling strategy for the IMM can be summarized by Algorithm 2[16].

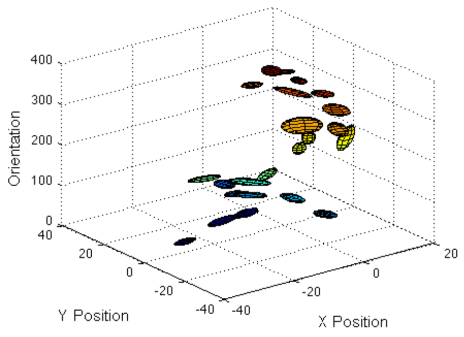
We would like our samples from our Gibbs sampler to be distributed *i.i.d* given the distribution, however because of the Markovian property of the sampling chain, we have to ensure that the chain has "mixed" well. A common measure of estimating the mixing state of the chain is to plot the autocovariance of various parameters of the model. For the semantic distributions learned, mixing seemed to occur within the first 1000 samples (spaced equally 10 samples apart) from the chain. We then use 100 samples equally spaced throughout the posterior to approximate the distribution - see Rasmussen for the full implementation and [16] for an extended discussion on the IGMM.

Once we have obtained our 100 independent samples from the posterior, we can form the probability distribution from which we can generate samples for a probabilistic planner. Running the IGMM on the circle dataset and extracting one sample corresponding to a maximal modal estimate of k_{rep}, we can render the 1-σ bounds of the mixture model and compare it to the data. Figure 3 shows the mixture model with $k_{rep} = 23$ components.

The IGMM is a robust way of estimating densities from

Algorithm 2 Gibbs sampling for the IMM

```
1: procedure GIBBS($numSamples$)
2:     for sample in $numSamples$ do
3:         for $n \in N$ do
4:             Sample indicators $c_n$ according to 19 and 20.
5:         end for
6:         Update $k_{rep}$ (# represented mixtures)
7:         for $k \in k_{rep}$ do
8:             Update $n_j$, the indicator count of mixture $j$
9:             Update weights $\pi_j = \frac{n_j}{n+\alpha}$
10:            Update unrepresented weights $\pi = \frac{\alpha}{n+\alpha}$
11:        end for
12:        for $k \in k_{rep}$ do
13:            Sample $\mu_j \sim p(\mu_j \mid \mathbf{c}, \mathcal{D}, s_j, \lambda, r)$
14:            Sample $s_j \sim p(s_j \mid \mathbf{c}, \mathcal{D}, \mu_j, \beta, w)$
15:        end for
16:        Update hyper-parameters:
17:        Sample $\lambda \sim p(\lambda \mid \mu, \gamma)$
18:        Sample $\gamma \sim p(\gamma \mid \mu, \lambda)$
19:        Sample $w \sim p(w \mid \mathbf{s}, \beta)$
20:        Sample $\beta \sim p(\beta \mid \mathbf{s}, w)$
21:    end for
22: end procedure
```



(a)

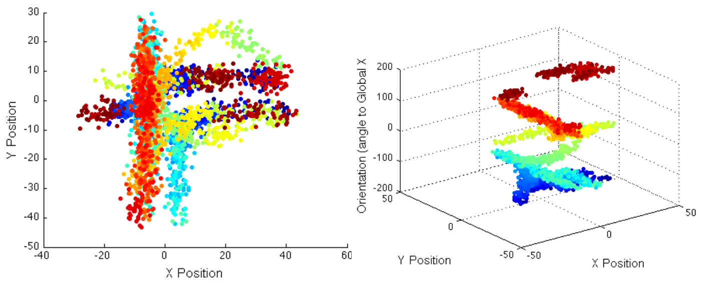
Fig. 3: The circle dataset from Figure 2 as approximated by a modal sample from the IGMM posterior of 23 guassians.

complex, high-dimensional data. However, in some cases, the use of an underlying parametric distribution (the Gaussian) causes the IGMM to fail at estimating densities that have a strong manifold structure. For example, consider an alternative dataset obtained from trajectories around (and through) a 4-way intersection shown in Figure 4.

In this dataset, the data tends to populate manifolds of the input-space, which is intuitive given the lack of variability in behavior as cars transition through intersections. We need to exploit the manifold structure so that the density estimation of this data is still possible. As such, we seek some form of dimensionality reduction in order to better analyze the underlying probability distribution.

IV. Learning in an appropriate dimension

Many techniques exist for dimensionality reduction, including linear methods (Principal Component Analysis, multi-dimensional scaling (MDS)) and non-linear techniques (Kernel-based PCA, ISOMAP). We make use of a non-linear technique - Locally Linear Embedding (LLE) [17] - to perform the dimensionality reduction, a technique which guarantees convergence on the global optimum and in addition enables us to perform reprojection of low-dimensional samples in feature-space back into input-space.



(a) (b)

Fig. 4: Observed data $\mathcal{D}$ around/through an intersection, as viewed over $\mathbf{x}$ and $\mathbf{y}$. This dataset exhibits a distinctly manifold structure, and as such the IGMM has difficulty estimating the density.

A. Locally-Linear Embedding

LLE attempts to compute a low-dimensional embedding from a high-dimensional input space such that nearby points in the input space remain co-located in the low-dimensional embedding. We seek to map our training data $\mathcal{D}$, consisting of $n = |\mathcal{D}_{i=1:|\mathcal{D}|}|$ input points from $\mathbb{R}^4$ to some embedded dimenson $\mathbb{R}^d$. The procedure for LLE is as follows:

1) For each data tuple $t_i = \langle x, y, \theta, |v| \rangle$ (of n such tuples), k nearest neighbors are selected, measured by some metric ρ (typically Euclidean distance)
2) The *reconstruction* weights (the weights required to reconstruct the datapoint from its neighbors) are calculated as follows:

$$\mathbf{E}(\mathbf{w}) = \arg\min_{w} \sum_{i=1}^{n} (t_i - \sum_{j=1}^{k} w_{ij} t_j)^2 \qquad (21)$$

This cost function is minimized for each datapoint in the input space, subject to the constraints that only the k-nearest neighbors of $\mathbf{t}_i$ are used. Saul et al. note that the weights for each point are invariant to translations, rescalings and rotations relative to its neighbors.
3) The final step of the algorithm is to generate the low-dimensional embedding. This is accomplished by choosing the d-dimensional coordinates of each output point y_i to minimize the embedding cost function:

$$\mathbf{y} = \sum_{i} (\mathbf{y}_i - \sum_{j} w_{ij} \mathbf{y}_j)^2 \qquad (22)$$

This is analogous to Equation 21, except that now the weights are held fixed, and the optimization proceeds over the outputs $\mathbf{y}$.

The only free parameters of the algorithm are the desired embedding dimension, and the number of nearest neighbors in Step 1 of the algorithm. The embedding dimension of the data can be estimated by maximum-likelihood, and the parameter k by a simple discrete search. The advantage of LLE is that the process can be readily reversed. We follow the method proposed by Wang et al.[18] whereby the weights for some newly-synthesized datapoint $\mathbf{z}$ in feature-space are recalculated according to:

$$\mathbf{E}(\mathbf{w}) = \arg\min_{w} \sum_{i=1}^{N} (\mathbf{z} - \sum_{j=1}^{k} w_{ij} \mathbf{y}_j(\mathbf{z}))^2 \tag{23}$$

where $\mathbf{y}(\mathbf{z})$ are the closest neighbors (in the original dataset) for the new point $\mathbf{z}$ in the embedding. This is essentially is the reverse of Step 1 of the LLE algorithm. The new tuple in input space $\hat{\mathbf{t}}$ is then generated by:

$$\hat{\mathbf{t}} = \sum_{j=1}^{k} w_j \mathbf{t}_j(\mathbf{z}) \tag{24}$$

where $\mathbf{t}(\mathbf{z})$ are the high-dimensional input tuples corresponding to the k-neighbors in the embedding. This simple procedure allows us to generate new data in the feature space that is representative of the original training data, however it does make various assumptions about the manifold in input space - we assume that our newly synthesized data are well approximated by their neighbors in input-space and that the manifold varies smoothly. In practice the newly generated data are a good approximation to the training data.

We now apply this dimensionality reduction and reprojection technique to the intersection dataset in Figure 4. The ML estimate for the intrinsic dimension of this data was ≈2, and the associated embedding - with 7 nearest neighbors - is depicted in $\mathbb{R}^2$ by Figure 5 (henceforth known as the *Bolshoi* embedding).

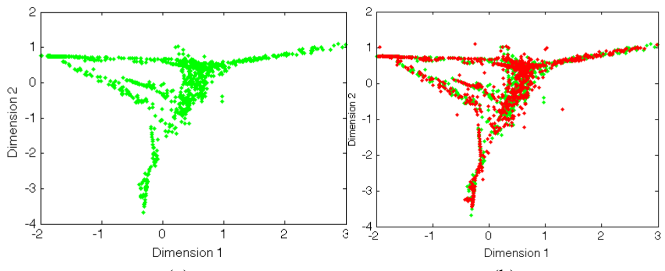


Fig. 5: The resultant embedding produced in $\mathbb{R}^2$ after running LLE on the intersection dataset (Figure 4) with $k = 7$ nearest neighbors(a). (b) depicts a newly synthesized feature-space dataset (shown in red) overlaid with the original embedding (shown in (a)) after running the IMM in $\mathbb{R}^2$. As can be seen, the generated samples are a good representation of the embedding.

We then implement the IGMM in this low dimensional embedding to produce the density estimate. Figure 5(b) shows 1000 samples generated from the posterior IGMM estimate of the density in $\mathbb{R}^2$ overlaid on the original embedding, with Figure 6 comparing the original dataset (a) and the newly synthesized dataset in input space (b).

V. Using the Learned Distributions

Once we can generate samples, either directly from the high-dimensional space of the distribution or via the embedding-reprojection approach, we need to incorporate the sampling strategy into the RRT algorithm. Planning in $\mathbb{SE}(2)$ requires generating random samples in (x, y, θ), which we can accomplish by marginalizing our learned distribution over the speed variable. Algorithm 3 illustrates the procedure for the semantically-biased RRT:

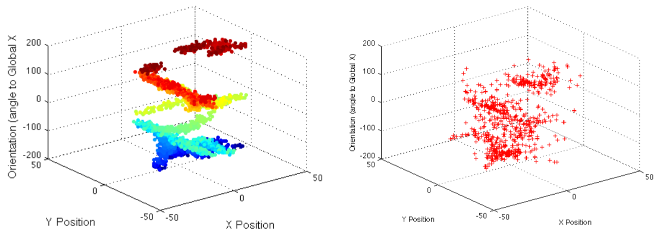


Fig. 6: A comparison of the original data (a), against the newly synthesized input-space dataset after performing the lifting from feature space into the input space (b)

Algorithm 3 Semantic-field based RRT Algorithm

```
1:  Run Algorithm 2 offline to obtain GMM estimate.
2:  procedure GENERATE_SF_RRT($x_{init}, K, \Delta t$)
3:      $\mathcal{T}.init(x_{init})$
4:      for $k \leftarrow 1, K$ do
5:          $x_{rand} \leftarrow SampleFromMM(1)$;        ▷ Alg. 1
6:          $x_{near} \leftarrow NEAREST\_NEIGHBOR(x_{rand}, \mathcal{T})$
7:          $u \leftarrow SELECT\_INPUT(x_{rand}, x_{near})$
8:          $x_{new} \leftarrow NEW\_STATE(x_{near}, u, \Delta t)$
9:          $\mathcal{T}.add\_vertex(x_{new})$;
10:         $\mathcal{T}.add\_edge(x_{near}, x_{new}, u)$;
11:     end for
12:     return($\mathcal{T}$)
13: end procedure
```

We do not have direct access to the weights π of the mixture model, so instead we sample according to $\frac{n_i}{N}$ which is the proportion of samples for each mixture as a fraction of all the training samples. If we have learned the distribution exactly in the high-dimensional space, then generating samples according to Line 5 in Algorithm 3 proceeds as in Algorithm 1. If instead we have been required to learn the density in some lower-dimensional embedding, we sample from the set of Gaussian mixtures in the lower space according to Algorithm 1, and then use the training data to lift them into the input-space at run-time via Equations 23 and 24.

VI. Results

The concept was tested on a standard planning task, generating trajectories for a planar vehicle in $\mathbb{SE}(2)$ in addition to a manipulator task. We show that in both cases, the paths produced by the semantically-biased planner are more similar to the expert paths than the conventional RRT solutions.

A. Speed profiling

As we have learned the distribution over $\langle \mathbf{x}, \mathbf{y}, \mathbf{\Theta}, |\mathbf{v}| \rangle$ data for the $\mathbb{SE}2$ task, we can easily obtain the speed profiles of the resultant solution path. This is done by evaluating the posterior distribution conditioned on the nodes in the tree:

$$p(|v| \mid \mathbf{x}, \mathbf{y}, \mathbf{\Theta}) \tag{25}$$

for each (x, y, θ) tuple in the RRT. Once we have run the IGMM, we can readily extract the velocity profiles from the conditional distributions of each node. Figure 7 shows the conditional distribution for one of the nodes in the final path.

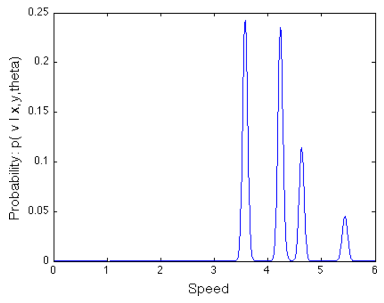


Fig. 7: Conditional distribution $p(|\mathbf{v}| \mid \mathbf{x}, \mathbf{y}, \theta)$ for one of the nodes in the resultant path. We calculate the expectation of this distribution to give the resultant speed.

Visible from Figure 7 are the contributions from various mixtures in the mixture model. To obtain the desired speed s for a node parameterized by x_i, y_i, θ_i in the solution path, we calculate the expectation of this conditional distribution:

$$s = \mathbb{E}[p(|\mathbf{v}| \mid x_i, y_i, \theta_i)] \tag{26}$$

An illustration of the speed profiles for one of the test runs is shown in Figure 9.

B. Path generation

1) Vehicle planning: Using the semantically-biased planning technique, a number of $\langle start, goal \rangle$ queries were solved for the circle dataset. Figure 8 contrasts the paths obtained by conventional RRT methods (green) against those obtained with a semantically-biased tree (blue) for a single $\langle start, goal \rangle$ instance:

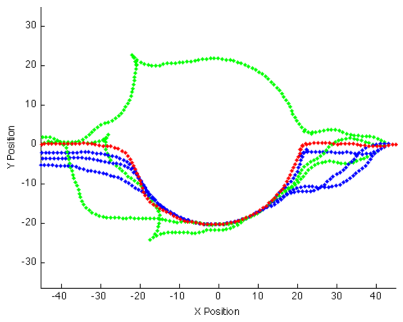


Fig. 8: A comparison of standard RRT solutions to the traffic circle problem (green) and the semantic-field based solutions (blue) with ground-truth data (shown in red) transitioning from a start state at $[50, 0]$ to a goal state at $[-50, 0]$.

We can again condition on the final path to obtain the speed profile for the resultant solution path. Figure 9 shows the resultant speed profile for one of the semantically-biased solutions shown in Figure 8.

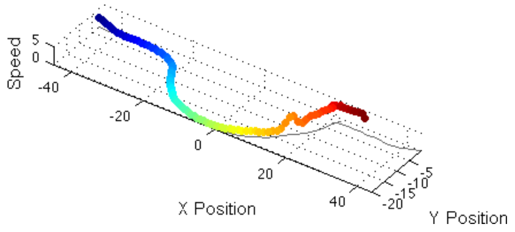


Fig. 9: Speed profile for one of the semantic-field solutions to the planning query as shown in Figure 8. This speed profile is representative of how the model was trained: the entry into the traffic circle was relatively high, with a low constant speed until the exit.

2) Manipulator planning: The process was also validated on a manipulator-planning problem for a standard platform, the Puma 560. Using the Robotics Toolbox developed by [19] an illustrative obstacle-avoidance problem was formulated. Shown in Figure 10(a) below is the initial setup for the task, with the goal being to move the end-effector from the initial position to the goal position (marked in green) over the top of the obstacle:

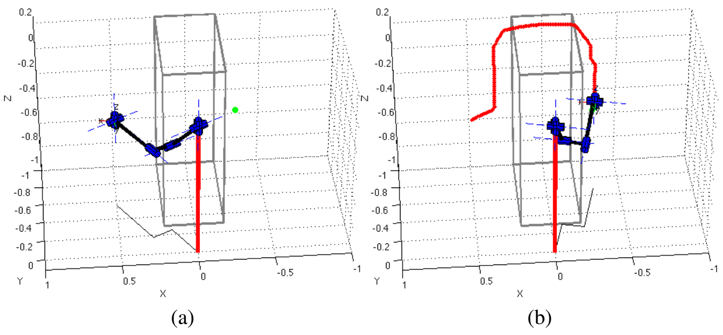


Fig. 10: Problem setup for the manipulator task. The goal is to move the end-effector from the start position (shown in (a)) to the end-location (denoted by the green marker) over the top of the box obstacle (shown in grey). An expert solution is shown in (b).

Figure 10(b) shows one of the training instances as demonstrated by the "expert". 10 such runs were obtained, and a density estimate was obtained using the IGMM. The performance of the semantically-biased planner was compared against a standard RRT solution, as shown in Figure 11, which depicts the semantically-biased solutions (blue) against the standard RRT solutions (green). As can be seen, the semantic solutions are more similar to the expert paths than those of the standard RRT.

C. Analysis

In order to compare the solutions obtained with different planning strategies, we require a metric that can meaningfully compare trajectories with differing scales and lengths. *Dynamic Time Warping* (DTW)[20] is a signal-analysis technique for comparing signals with differing offsets and

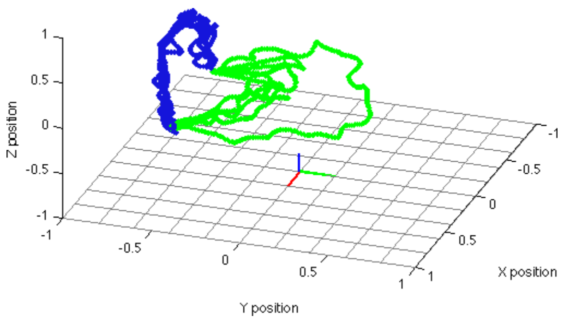


Fig. 11: Solutions to the manipulator task of both the semantically-biased planner (blue) and a standard RRT solution (green). As can be seen, the semantic-planner produces paths more similar to those of the expert - the semantic planner has learned to plan “over” the box.

component shapes. Given two data series, $\mathbf{x} \in (x_1, \ldots, x_N)$ and $\mathbf{y} \in (y_1, \ldots, y_N)$, the overall distortion between the two signals $\mathcal{D}(\mathbf{x}, \mathbf{y})$ is based on a sum of local distances between elements $d(x_i, y_i)$. Any alignment warping Φ aligns the two sequences with a point-to-point mapping $\Phi = (\Phi_x, \Phi_y)$ with length K_Φ:

$$\mathbf{x}_{\Phi_x}(k) \Leftrightarrow \mathbf{y}_{\Phi_y}(k) \quad 1 \leq k \leq K_\Phi \tag{27}$$

with the optimal alignment minimizing the distortion. This is a minimization problem:

$$\mathcal{D}_\Phi(\mathbf{x}, \mathbf{y}) = \min_\Phi \frac{1}{K_\Phi} \sum_{k=1}^{K_\Phi} d(\mathbf{x}_{\Phi_t}(k), \mathbf{y}_{\Phi_r}(k)) \tag{28}$$

that can be solved with a dynamic-programming approach, given end-point constraints and one for monotonicity. We can therefore use DTW to calculate the minimum distance between two signals, and use it as a criteria for comparing planned paths with the ground-truth data (paths observed from a human expert). Table I compares the average distances (measured by DTW) of 10 RRT and semantic-field solutions to a ground-truth trajectory for the problem shown in Figure 8:

TABLE I: Path distances to ground-truth: $\mathbb{SE}2$ problem

Planner Type	Mean Distance to ground truth
Semantic Field	0.11
Conventional RRT	0.48

This confirms our intuition that we can use expert data to teach an RRT to plan in a way akin to humans. Implicitly in this paper we have produced a framework capable of learning arbitrary sampling policies with no need for *a-prior* parameterization.

VII. Conclusion

We have presented a novel approach to learn sampling distributions for stochastic motion planners. By utilizing a non-parametric approach to density estimation around pertinent features, we can learn approximating distributions and incorporate these distributions into an RRT to produce natural paths. In the event that the input space is an over-parameterization of the data, we use a dimensionality reduction technique in order to perform density estimation in the implicit dimension of the data. It is apparent that incorporating semantic information leads to more natural paths that are similar to those we have observed under expert control, and we assume this technique will be effective in other planning related scenarios.

VIII. Acknowledgements

This work was funded by the Office of Naval Research (ONR) Grant N00140810337. The authors would like to thank the reviewers for their comments.

References

[1] S. M. Lavalle, “Rapidly-exploring random trees: A new tool for path planning,” Tech. Rep., 1998.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” in *IEEE International Conference on Methods and Models in Automation and Robotics, 2005*. MorganKaufmann, 1997, pp. 566–580.

[3] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, “Planning collision-free reaching motions for interactive object manipulation and grasping,” *Eurographics*, vol. 22, pp. 313–322, 2003.

[4] J. Cortes and T. Simeon, “Sampling-based motion planning under kinematic loop-closure constraints,” in *In 6th International Workshop on Algorithmic Foundations of Robotics*. Springer-Verlag, 2004, pp. 59–74.

[5] J. Kim and J. Ostrowski, “Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints,” in *Robotics and Automation, 2003. Proceedings. ICRA ’03. IEEE International Conference on*, vol. 2, Sept. 2003, pp. 2200–2205 vol.2.

[6] Z. Sun, I. Member, D. Hsu, T. Jiang, H. Kurniawati, J. H. Reif, and I. Fellow, “Narrow passage sampling for probabilistic roadmap planning,” 2005.

[7] V. Boor, M. H. Overmars, and A. F. van der Stappen, “The gaussian sampling strategy for probabilistic roadmap planners,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1018–1023 vol.2. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1999.772447

[8] Y. Yang and O. Brock, “Adapting the sampling distribution in prm planners based on an approximated medial axis,” 2004.

[9] N. M. Amato, O. B. Bayazit, and L. K. Dale, “Obprm: An obstacle-based prm for 3d workspaces,” 1998.

[10] M. Zucker, J. Kuffner, and J. A. D. Bagnell, “Adaptive workspace biasing for sampling based planners,” in *Proc. IEEE Int’l Conf. on Robotics and Automation*, May 2008.

[11] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, “Motion planning for urban driving using rrt,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept. 2008, pp. 1681–1686.

[12] K. Macek, M. Becked, and R. Siegwart, “Motion planning for car-like vehicles in dynamic urban scenarios,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 4375–4380.

[13] R. Neal, “Density modeling and clustering using Dirichlet diffusion trees,” *Bayesian Statistics*, vol. 7, pp. 619–629, 2003.

[14] R. Adams, I. Murray, and D. MacKay, “The Gaussian process density sampler,” *Advances in Neural Information Processing Systems*, vol. 21, 2009.

[15] C. E. Rasmussen, “The infinite gaussian mixture model,” in *In Advances in Neural Information Processing Systems 12*. MIT Press, 2000, pp. 554–560.

[16] T. Chen, J. Morris, and E. Martin, “Probability density estimation via an infinite gaussian mixture model: application to statistical process monitoring,” *Journal Of The Royal Statistical Society Series C*, vol. 55, no. 5, pp. 699–715, 2006. [Online]. Available: http://econpapers.repec.org/RePEc:bla:jorssc:v:55:y:2006:i:5:p:699-715

[17] Roweis, T. Martinetz, K. Schulten, N. Netw, V. Kumar, A. Grama, A. Gupta, and G. Karypis, “Nonlinear dimensionality reduction by locally linear embedding,” 2000.

[18] J. Wang, M. Xu, H. Wang, and J. Zhang, “Classification of imbalanced data by using the smote algorithm and locally linear embedding,” in *Signal Processing, 2006 8th International Conference on*, vol. 3, 16-20 2006, pp. –.

[19] P. Corke, “A robotics toolbox for MATLAB,” *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, Mar. 1996.

[20] H. Strik and L. Boves, “Averaging physiological signals with the use of a DTW algorithm,” *Proceedings SPEECH*, vol. 88, no. 7, pp. 883–890, 1988.