CRF-Filters: Discriminative Particle Filters for Sequential State Estimation

Benson Limketkai Dieter Fox Lin Liao
Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195
{bensonl,fox,liaolin}@cs.washington.edu

Abstract— **Particle filters have been applied with great success to various state estimation problems in robotics. However, particle filters often require extensive parameter tweaking in order to work well in practice. This is based on two observations. First, particle filters typically rely on independence assumptions such as "the beams in a laser scan are independent given the robot's location in a map". Second, even when the noise parameters of the dynamical system are perfectly known, the sample-based approximation can result in poor filter performance. In this paper we introduce CRF-Filters, a novel variant of particle filtering for sequential state estimation. CRF-Filters are based on conditional random fields, which are discriminative models that can handle arbitrary dependencies between observations. We show how to learn the parameters of CRF-Filters based on labeled training data. Experiments using a robot equipped with a laser range-finder demonstrate that our technique is able to learn parameters of the robot's motion and sensor models that result in good localization performance, without the need of additional parameter tweaking.**

I. Introduction

Estimating the state of a dynamical system is of fundamental importance in robotics. Particle filters are a sample-based implementation of Bayes filters, which are generative models that recursively estimate posterior probability distributions over the state of a dynamical system [14], [4]. Due to their expressiveness and conceptual simplicity, particle filters have become extremely popular for state estimation in robotics. Successful applications of particle filters include robot localization, mapping, and people tracking.

While the sample-based representation gives particle filters a huge flexibility in modeling dynamical systems, this representation comes at the cost of increased computational complexity. Furthermore, to make particle filters work in practice, one often has to resort to extensive manual tweaking of filter parameters. This has two main reasons: First, in order to generate good filter performance, the parameters must model both the noise of the dynamical system and the approximation error due to the limited number of particles. As a result, for instance, even when the noise of the system is perfectly known, a particle filter solely modeling this noise might perform poorly especially when posteriors are highly peaked [14]. To alleviate this problem, it is common practice to artificially inflate the noise of the system, thereby "smoothing" posterior distributions [4].

The second potential reason for poor particle filter performance is due to invalid independence assumptions underlying the probabilistic model. For instance, a very common assumption made by Bayes filters is the so-called naive Bayes assumption for measurement vectors. In the context of robot localization, this assumption states that the individual beams of a sonar or laser range-scan are independent given the robot's location [5], [6], [14]. While such an assumption significantly reduces the complexity of measurement models, it is often violated and causes overly peaked likelihood functions, even when the model parameters are learned from real data. A standard way of dealing with this problem is to artificially reduce the gain of the learned sensor model, for instance by applying an exponential smoothing coefficient to each individual observation component [14].

In this paper we introduce CRF-Filters, a novel variant of particle filters. CRF-Filters build on conditional random fields (CRF), which are discriminative probabilistic models designed to handle high-dimensional and correlated observations [7], [12]. CRF-Filters extend CRFs to continuous domains by performing particle filtering during inference and learning. By learning model parameters discriminatively, CRF-Filters automatically learn parameters that maximize filter performance, taking dependencies, sensor noise, and sample-based approximations into account. For instance, our experimental evaluation of laser-based robot localization shows that CRF-Filters learn different parameters depending on how many beams are contained in a laser scan and depending on whether the robot performs global localization or position tracking.

This paper is organized as follows. After discussing related work in the next section, Section III describes generative Bayes filters and CRF-Filters. In this section we also show how to model a wide range of transition and measurement models in our framework. Sample-based inference and discriminative learning is described in Section IV, followed by experimental results. We conclude in Section VI.

II. Related Work

Particle filters have been applied successfully to a variety of estimation problems in robotics. However, it is well known that particle filters require large numbers of particles especially when the observation noise is very small. Several researchers showed how to deal with this problem by generating particles from an improved proposal distribution [15], [16]. Recently, Pfaff and colleagues [9] showed how to adapt

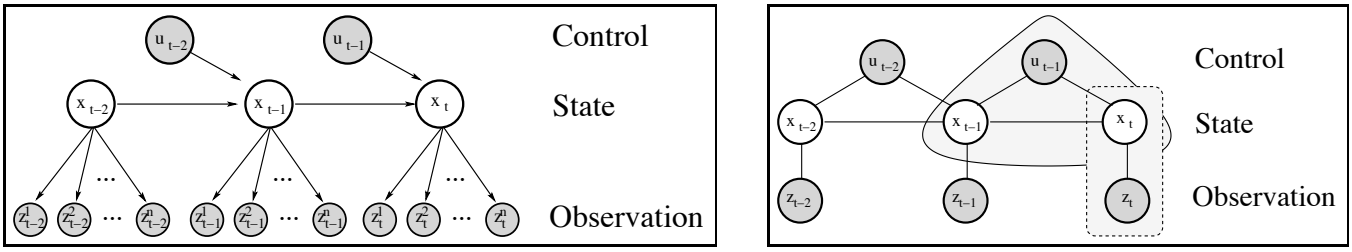

Fig. 1. (left) Directed graphical model of a Bayes filter with typical naive Bayes independence assumption on observations. (right) Undirected graphical model of a CRF-Filter. Light grey areas indicate prediction clique (solid) and correction cliques (dashed) for time t.

the smoothness of a sensor model depending on the density of the particles during robot localization. This approach attempts to model the additional sensor noise needed to incorporate the sample-based approximation of the posterior. While these approaches greatly improve the efficiency of the resulting filters, they do not address the problem of how to learn the parameters of the filters. Furthermore, they provide no foundation for handling the dependencies inherent in high-dimensional sensor data such as laser range-scans.

Abbeel and colleagues [1] showed how to discriminatively train the noise parameters of Kalman filters. They demonstrated that discriminatively trained Kalman filters outperform their generatively trained counterparts. [2] and [10] showed that consistent motion and observation models can be learned for mobile robots. At the current stage of their work, however, it is not clear how their fully unsupervised techniques can scale to more realistic problems in sequential state estimation.

Taycher and colleagues [13] used conditional random fields for people tracking. They applied standard CRF models by discretizing the continuous state space. While such an approach worked well in their specific application, grid-based representations do not scale well to higher dimensional state spaces. To overcome this limitation, we use sample-based approximations for inference and learning in our CRF-Filters. The resulting particle filter algorithm is a special version of nonparametric belief propagation introduced by [11]. While their focus is on sample-based inference in generative models such as Markov random fields, we show how particle filtering can be used for both inference and discriminative learning of model parameters in CRFs.

III. CRF-Filters

In this section we show how to apply conditional random fields to sequential state estimation. We start with a brief description of Bayes filters, a generative framework for sequential state estimation (see [14] for details).

A. Bayes Filters for Sequential State Estimation

The goal of Bayesian filtering is to estimate the posterior over the state, $\mathbf{x}_t$, of a dynamical process, conditioned on all control information, $\mathbf{u}_{1:t-1}$, and sensor measurements, $\mathbf{z}_{1:t}$, obtained through time t. Typically, time is discretized, but all random variables can be continuous. In robot localization, for instance, $\mathbf{x}_t$ describes the robot's $\langle x, y, \theta \rangle$ location at time t, $\mathbf{u}_{t-1}$ is odometry information about the robot's motion between time $t-1$ and t, and $\mathbf{z}_t$ is a sensor measurement such as a laser range-scan.

Bayes filters are generative models that can be described via the directed graphical model shown in the left panel of Figure 1. The model defines the following recursive equation to compute posterior distributions over the hidden state $\mathbf{x}_t$:

$$p(\mathbf{x}_t|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t}) \propto$$
$$p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{u}_{t-1}, \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t-2}, \mathbf{z}_{1:t-1})\, d\mathbf{x}_{t-1} \quad (1)$$

The term $p(\mathbf{x}_t|\mathbf{u}_{t-1}, \mathbf{x}_{t-1})$ is a probabilistic model of the system dynamics, which predicts the state at time t based on the previous state and control information. As can be seen, Bayes filters rely on Bayes rule to model posterior distributions. A key component of such a filter is the generative measurement model $p(\mathbf{z}_t|\mathbf{x}_t)$, which describes how measurements are generated from the hidden state. Very often, observations $\mathbf{z}_t$ are measurement vectors $(z_t^1, z_t^2, \ldots, z_t^n)'$, such as a camera image or a scan of laser range-readings. In order to avoid modeling high-dimensional distributions over such vectors, Bayes filters typically make the naive Bayes like assumption that the individual readings z_t^i are independent given the state $\mathbf{x}_t$ [14]. The resulting likelihood model, indicated by the multiple arcs in Figure 1, is then given by

$$p(\mathbf{z}_t|\mathbf{x}_t) = \prod_{i=1}^{n} p(z_t^i \mid \mathbf{x}_t)\ . \quad (2)$$

However, since such likelihood models are learned for each measurement component independently, the resulting likelihood can become overly peaked.

B. CRF-Filters for Sequential State Estimation

Conditional random fields are undirected graphical models that were developed for labeling sequence data [7] (see [12] for a recent overview). The nodes in a CRF represent random variables, just like those in the directed graphical model underlying Bayes filters. The main difference, however, is that the connections between nodes are undirected and the dependencies between nodes are not restricted to normalized, conditional probabilities, but can be any non-negative function. Furthermore, instead of relying on Bayes rule to estimate distributions over hidden states from observations, CRFs directly represent the conditional distribution over hidden states given the observations. CRFs thereby avoid the need for generative measurement models, which gives them substantially more modeling flexibility.

The distribution over hidden states of a CRF is defined via the nodes and the clique structure imposed by the edges

between them. To see, consider the structure of our CRF-Filter shown in the right panel in Figure 1. For each time step t, the model has two cliques (fully connected sub-graphs) consisting of $(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ and $(\mathbf{x}_t, \mathbf{z}_t)$, respectively. For this structure, the conditional distribution over the hidden states $\mathbf{x}_{0:T}$ of a sequence of length $T+1$ factorizes into the following product of *clique potentials*, which are non-negative functions defined over the two types of cliques:

$$p(\mathbf{x}_{0:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{0:T-1}) = \frac{1}{Z(\mathbf{z}_{1:T}, \mathbf{u}_{1:T-1})} \cdot \prod_{t=1}^{T} \phi_p(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1})\ \phi_m(\mathbf{x}_t, \mathbf{z}_t) \quad (3)$$

$Z(\dots) = \int \prod_{t=1}^{T} \phi_p(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \phi_m(\mathbf{x}_t, \mathbf{z}_t)\ \ d\mathbf{x}_{1:T}$ is the normalizing partition function. It is necessary since the potential functions in CRFs do not need to be normalized, in contrast to the conditional probabilities used in Bayes filters. Since the computation of this partition function requires integration over all possible configurations of the continuous hidden states, exact inference is not possible in CRF-Filters.

ϕ_p in (3) is the *prediction potential*. It models the temporal evolution of the process and is closely related to the dynamics model in the Bayes filter formulation (1). ϕ_m, the *measurement potential*, corresponds to the observation model in the Bayes filter. Intuitively, the potentials capture the "compatibility" among the variables in the clique; the larger the potential value, the more likely the configuration.

Following standard CRF convention, we describe clique potentials ϕ by log-linear combinations of *feature functions*. For instance, $\phi(\mathbf{x}_1, \mathbf{x}_2) = \exp\{\mathbf{w}' \cdot \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2)\}$ defines a potential for variables $\mathbf{x}_1$ and $\mathbf{x}_2$, where $\mathbf{f}$ is a vector of features extracted from the variables, and $\mathbf{w}$ is a weight vector (Section IV describes how to learn weights). The individual components of the feature vectors are typically real-valued or binary functions. CRFs have been applied successfully to problems involving thousands of features [12].

C. Application to Robot Localization

We will now show how CRF-Filters can be used for mobile robot localization using proximity sensors such as sonar sensors or laser range-finders. Here, the state $\mathbf{x}_t = (x_t, y_t, \theta_t)$ represents a robot's location and orientation in a given map of the environment.

Prediction potential for odometry motion model

The prediction potentials ϕ_p model the temporal evolution of the dynamical process. Each $\phi_p(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ measures the compatibility between two consecutive states, $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, and the control information $\mathbf{u}_{t-1}$. For instance, consider the following odometry motion model commonly used in mobile robotics. Here, the robot's motion between time $t-1$ and t is estimated using wheel encoders, and the vector $\mathbf{u}_t = (\delta_{\text{rot1}}, \delta_{\text{trans}}, \delta_{\text{rot2}})'$ is specified by an initial rotation of δ_{rot1} degrees, followed by a translation of δ_{trans} cm, and a final rotation of δ_{rot2} degrees [14].

We can use prediction potentials to specify a Gaussian noise model on the individual components of $\mathbf{u}_t$, as is often adopted by particle filter and Kalman filter models [14]. To do so, we define the feature functions of prediction potentials ϕ_p as

$$f_p^i(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \quad := \quad (u_{t-1}^i - \hat{u}_{(\mathbf{x}_{t-1}, \mathbf{x}_t)}^i)^2, \quad (4)$$

where u_{t-1}^i denotes the i-th component of the odometry measurement vector $\mathbf{u}_{t-1}$. $\hat{\mathbf{u}}_{(\mathbf{x}_{t-1}, \mathbf{x}_t)}$ is the *derived odometry vector*, which is the odometry value that corresponds to noise-free motion between state $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$. Each component of the feature vector $\mathbf{f}_p$ is thus the squared difference between the measured and the derived odometry vectors [1]. The corresponding prediction clique potential is

$$\phi_p(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \exp\left\{\mathbf{w}_p' \cdot \mathbf{f}_p(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1})\right\}, \quad (5)$$

which defines a quadratic penalty function that corresponds to an (unnormalized) Gaussian noise model. More specifically, let w_p^i be the i-th component of the weight vector $\mathbf{w}_p$. It can be shown that (5) represents a Gaussian noise model with mean at the odometry measurement u_t^i and variance $\sigma^2 = 1/(-2w_p^i)$ (we constrain these weight values to be negative). This connection to Gaussian noise models will turn out to be very useful for sampling-based inference, as described in Section IV-A.

Measurement potential for proximity sensor model

Proximity sensors such as sonar sensors or laser range-finders measure distances to obstacles around the robot. Mobile robots are often equipped with multiple sonar sensors and/or a laser range-finder that typically generates 180 beams of distance measurements per planar scan. To develop appropriate feature functions for the measurement potentials, we show how to model a slightly simplified version of the generative, beam-based approach given in [5], [14]. This model will also enable us to compare our CRF-Filter with an established, generative Bayes filter technique. We first describe the generative version of the sensor model.

Here, the likelihood $p(z_t^i|\mathbf{x}_t)$ of a specific sensor beam is computed by first determining the expected measurement, $\hat{z}_t^i$, given the robot location $\mathbf{x}_t$. This measurement is typically determined by ray-tracing in a map of the environment, starting at the sensor location (determined by $\mathbf{x}_t$). The generative model applies the following mixture distribution.

$$p(z_t^i \mid \mathbf{x}_t) \quad = \quad \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\max} \\ \alpha_{\text{rand}} \end{pmatrix}' \cdot \begin{pmatrix} \mathcal{N}(z_t^i; \hat{z}_t^i, \sigma_{\text{hit}}^2) \\ I(z_t^i = z_{\max}) \\ \frac{1}{z_{\max}} \end{pmatrix} \quad (6)$$

Here, $z_{\max}$ is the maximum range of the sensor. The first component, which models a measurement that successfully detects the obstacle in the map, is a Gaussian centered at the expected distance $\hat{z}_t^i$. The second component models maximum range measurements, and the third component models erroneous measurements, which are represented by a uniform distribution. The vector $(\alpha_{\text{hit}}, \alpha_{\max}, \alpha_{\text{rand}})'$ represents the weights of the mixture components.

In a typical generative Bayes filter, the likelihood of a sonar or laser scan is then given by multiplication of the

[1] To model noise that is relative to the size of the motion, we divide the individual components of $\mathbf{f}_p$ by values derived from $\mathbf{u}_{t-1}$, see [14].

individual beam likelihoods, as given in (2). The parameters of such a model are typically learned for individual beams using expectation maximization (EM), based on training data for which ground truth locations are known [14].

To represent a similar sensor model in our CRF-Filter, we define the binary value c_t^i, which is true if the difference between the measurement and the expected measurement is less than 20 centimeters (this models "correct" measurements). We also define the binary variable m_t^i, which is true if the measurement z_t^i is max range, and $\hat{\mathrm{m}}_t^i$, which is true if the expected measurement $\hat{z}_t^i$ is max range (that is, the closest obstacle in the sensing direction is further away than max range). The feature vector $\mathbf{f}_m^i$ modeling each sensor beam is then defined by considering the combinations of measured and expected max range measurements:

$$\mathbf{f}_m^i(\mathbf{z}_t, \mathbf{x}_t) \quad = \quad \begin{pmatrix} (\neg \mathrm{m}_t^i \wedge \neg \hat{\mathrm{m}}_t^i)\ c_t^i\ (z_t^i - \hat{z}_t^i)^2 \\ (\neg \mathrm{m}_t^i \wedge \neg \hat{\mathrm{m}}_t^i)\ \bar{c}_t^i \\ (\neg \mathrm{m}_t^i \wedge\ \ \hat{\mathrm{m}}_t^i) \\ (\ \ \mathrm{m}_t^i \wedge \neg \hat{\mathrm{m}}_t^i) \\ (\ \ \mathrm{m}_t^i \wedge\ \ \hat{\mathrm{m}}_t^i) \end{pmatrix} \tag{7}$$

The first case corresponds to the Gaussian component of the generative likelihood model. The second case models a random measurement (not unlike the third component of the generative model), and the other cases model different combinations of measured and expected maximum range measurements. The correction clique potential ϕ_m for this sensor model follows as

$$\phi_m(\mathbf{x}_t, \mathbf{z}_t) \quad = \quad \exp\left\{ \mathbf{w}_m' \sum_i^n \mathbf{f}_m^i(\mathbf{z}_t, \mathbf{x}_t) \right\}, \tag{8}$$

where $\mathbf{w}_m$ is the 5-dimensional weight vector modeling the impact of the different components of the feature vector (7).

IV. Inference and Discriminative Learning

A. Particle Filtering for CRF-Filters

As indicated in (3), CRFs estimate posterior distributions over complete state sequences $\mathbf{x}_{1:T}$, which corresponds to smoothing in the state estimation context. However, just as for Bayes filters, we can recursively estimate filtering posteriors in CRF-Filters. It turns out that the resulting particle filter algorithm is an instance of *nonparametric belief propagation*, which was introduced for sample-based inference in undirected Markov models [11]. Here, we omit a derivation and describe only the basic particle filtering algorithm for CRF-Filters, as given in Alg. 1.

As can be seen, the algorithm is virtually identical to generative particle filters, which update sample sets according to a procedure often referred to as sequential importance sampling with re-sampling [4]. The loop starting in Step 2 generates a weighted particle $\mathbf{x}_t^{(i)}$ for each particle $\mathbf{x}_{t-1}^{(i)}$ drawn via re-sampling in Line 1. In Step 3, the particle filter makes a prediction by sampling from the *marginal prediction potential*, rather than a conditional distribution, as is done in generative particle filtering. Given the specific form of prediction clique potentials defined in Section III-C, however, this marginal potential is an unnormalized Gaussian model

Inputs : Control $\mathbf{u}_{t-1}$, measurement $\mathbf{z}_t$, sample set $S_{t-1} = \{\langle \mathbf{x}_{t-1}^{(i)}, \alpha_{t-1}^{(i)} \rangle \mid i = 1, \ldots, N\}$

1 Re-sampling: Draw N samples $\mathbf{x}_{t-1}^{(i)}$ from S_{t-1} with probability proportional to importance weights $\alpha_{t-1}^{(i)}$;
2 **for** $i = 1, 2, \cdots, N$ **do**
3 Prediction: Sample $\mathbf{x}_t^{(i)} \sim \phi_p(\mathbf{x}_t, \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1})$;
4 Importance sampling: $\alpha_t^{(i)} = \phi_m(\mathbf{x}_t^{(i)}, \mathbf{z}_t)$;
5 **end**

Algorithm 1: Particle filtering for CRF-Filters

with a diagonal covariance matrix[2]. We can easily sample from this marginal by first generating a "noisy version", $\mathbf{u}_{t-1}^{(i)}$, of the control vector via sampling Gaussian noise for each control vector component, followed by the computation of the resulting $\mathbf{x}_t^{(i)}$. The Gaussian noise of each component is specified by the weight vector $\mathbf{w}_p$ in (5). Finally, Step 4 incorporates the measurement by weighting the particle by the correction clique potential ϕ_m.

B. Discriminative Parameter Learning

The parameter weights of CRFs are typically learned discriminatively using training sequences that contain the ground truth hidden states $\mathbf{x}_{1:T}^*$ along with $\mathbf{z}_{1:T}$ and $\mathbf{u}_{1:T-1}$. The most-widely used learning criterion for CRFs is to maximize the conditional log-likelihood of the training data, $\log p(\mathbf{x}_{1:T}^* \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T-1})$. This objective function is typically optimized using techniques such as L-BFGS [12].

Since computation of the weight gradients and conditional likelihoods can be challenging in continuous domain CRF-Filters, we resort to an adapted version of the averaged perceptron algorithm, which lends itself to a simple implementation. The averaged perceptron algorithm was introduced by Collins [3] for discriminative training of discrete state hidden Markov models in natural language processing. The key idea of this technique is to iteratively update the weights based on the difference between feature values computed from the ground truth sequence $\mathbf{x}_{1:T}^*$ and feature values given by the maximum aposteriori (MAP) sequence $\hat{\mathbf{x}}_{1:T} = \mathrm{argmax}_{\mathbf{x}_{1:T}} p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}, \mathbf{z}_{1:T})$. Our approach determines an approximate MAP sequence using the history of the most likely particle at the final step T of the filter. The overall learning algorithm is summarized in Alg. 2.

The algorithm takes as input a sequence of sensor and odometry measurements along with the ground truth locations. At each iteration, the algorithm picks a sub-sequence from the training data and performs particle filtering to compute the MAP location sequence $\hat{\mathbf{x}}_{1:t}$ (Step 3). This sequence and the ground truth locations are used to determine the values $\mathbf{f}(\hat{\mathbf{x}}_{1:T}, \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t})$ and $\mathbf{f}(\mathbf{x}_{1:t}^*, \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t})$, respectively. These values are computed by adding the feature vectors (4) and (6) over the complete sub-sequence, with locations instantiated to $\hat{\mathbf{x}}_{1:T}$ and $\mathbf{x}_{1:t}^*$. The difference Δ between these values provides the gradient for the weight

[2]Note that more general prediction potentials might require additional normalization factors for the different particles and more advanced sampling methods such as Markov chain Monte Carlo (MCMC).

Inputs : Controls $\mathbf{u}_{1:T-1}$, measurements $\mathbf{z}_{1:T}$, ground truth positions $\mathbf{x}^*_{1:T}$, initial weight vector $\mathbf{w}_0$

Output: Estimated weight vector $\mathbf{w}$

1 $\mathbf{w} = \mathbf{w}_0$;
2 **repeat**
3 Randomly pick sub-sequence and estimate MAP sequence $\hat{\mathbf{x}}_{1:t}$ using CRF-Filter with weights $\mathbf{w}$;
4 Compute feature difference $\Delta = \mathbf{f}(\mathbf{x}^*_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t}) - \mathbf{f}(\hat{\mathbf{x}}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t})$;
5 Initialize step size $\mu = 1$;
6 Randomly pick sub-sequence for testing;
7 **for** $i = 1, 2, \cdots, M$ **do**
8 $\mathbf{w}_{test} = \mathbf{w} + \mu\Delta$;
9 **if** *CRF-Filter loses track using* $\mathbf{w}_{test}$ **then**
10 $\mu = 0.5\ \mu$;
11 **else**
12 $\mathbf{w} = \mathbf{w}_{test}$; break ;
13 **end**
14 **end**
15 **until** $\mathbf{w}$ *converges* ;

Algorithm 2: Learning algorithm for CRF-Filters

update (note that this Δ is closely related to the weight gradient of the conditional log-likelihood function [12]).

An important question in this context is the step size used for the weight update. While too small updates lead to slow convergence, too large updates might lead to poor filter performance. In the loop starting at Step 7, the algorithm searches for an appropriate step size by testing a sequence of exponentially decreasing steps along the gradient (each update in Step 8 ensures that the components of the weight vector corresponding to noise variances remain negative). For each step size, the algorithm checks if the filter successfully localizes the robot on a randomly picked test sequence (Step 9). This check is rather straightforward using the ground truth locations in the test sequence. The weight update is accepted in Step 12 if the algorithm successfully tracks, otherwise a smaller step size is tested (Step 10). The algorithm stops when changes in $\mathbf{w}$ fall under a pre-specified threshold.

To summarize, our learning algorithm performs approximate gradient steps in the conditional log-likelihood of the training data, where step sizes are chosen to avoid weight vectors that result in filter divergence.

V. Experiments

We evaluated CRF-Filters in the context of mobile robot localization. To do so, we used a Pioneer2-DX robot equipped with a SICK laser range-finder. The laser provides scans containing 181 beams, covering a planar area of 180 degrees in front of the robot. To learn the parameters of the motion and sensor model, we moved the robot through our department building, collecting 1,312 laser scans and odometry mesurements over a path of length 360m. For testing, we collected traces of length 200m from a different environment. The map and ground truth locations for learning and evaluation were generated by using a highly-accurate scan-matcher to align the scans. Figure 2 shows the high-resolution maps we used for training and testing.

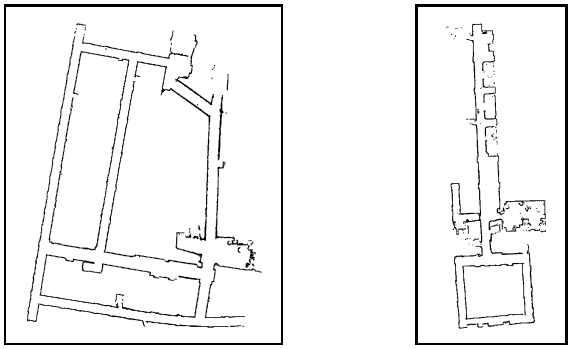

Fig. 2. Maps used for training (left) and testing (right).

Training of the motion and measurement model parameters typically converged after 90 iterations, each of which performed a particle filtering run to compute the gradient and three tests to verify that the new weights are still sufficient for tracking. Depending on the type of experiment, the particle filter was either initialized with a Gaussian distribution centered on the ground truth start location of the robot, or with a uniform distribution. Parameter learning typically took between 10 and 30 minutes (depending on the number of laser beams) for tracking and up to 10 hours for global localization on a standard desktop PC.

Comparison with generatively trained particle filter: We learned 10 sets of weights for tracking. For each set of weights, we ran 40 tracking tests, where each test consisted of starting the robot at a random point in the test trace. We then computed the average error of the mean of the particle set from the ground truth location over all 400 test traces. To learn the parameters of the generative particle filter, we used expectation maximization, as described in [14]. The generative model had an average error of 7.5cm, whereas the CRF-Filter had an average of 7.1cm.

For global localization, we learned 3 sets of weights. Similar to tracking, we ran 40 tests per set of weights. We then counted how often the particle filter was able to globally localize the robot using 25,000 particles. The generative model localized the robot 30% of the time. The highly-peaked nature of the generative model caused the particles to quickly concentrate in one area of the map–usually in the wrong area. Our CRF-Filter was able to localize the robot 96% of the time. The table below summarizes our results.

	Tracking Error (cm)	Global Localization Accuracy (%)
Generative	7.52 cm	30 %
CRF-Filter	7.07 cm	96 %

Dealing with dependent measurement vectors: Vanilla particle filters are brittle w.r.t. peaked observation likelihoods [4]. This problem typically occurs when high-dimensional observations are used, especially when the individual components of the obervations are learned independently of each other, as is typically done for generative approaches. In this experiment we investigated whether CRF-Filters are able to learn parameters that work in such

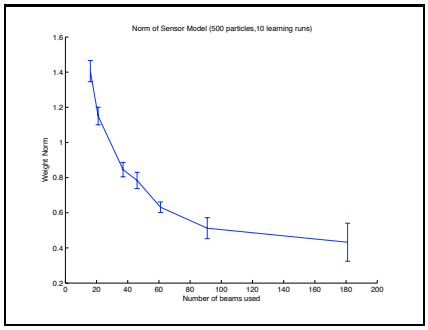

Fig. 3. Norm of sensor model weights as function of scan size.

situations. To do so, we used different numbers of the 181 laser beams, and learned model parameters for each number of beams. For training and testing, we used 500 particles initialized at the ground truth location.

To see how CRF-Filters behave for different scan sizes, we plot the norm of the sensor model weights learned for the different settings (see Figure 3). These weights indicate how peaked the measurement clique potential of each individual laser beam is. As can be seen, the model automatically learns smoother potentials (per beam) the more beams are used.

Considering particle uncertainty: In an additional experiment comparing weights learned for global localization and tracking we found that the average norm of the sensor model learned for global localization using 181 beams is 0.0852. This is much lower (resulting in a smoother sensor model) than the average of 0.4 when learning tracking parameters.

Theses experiments demonstrate that the CRF-Filter is able to deal with high-dimensional observations and automatically compensates for overly peaked likelihoods, something that is typically done manually for generative particle filters.

VI. Conclusions

We presented CRF-Filters, a novel, discriminatively trained approach to particle filtering. CRF-Filters are continuous versions of conditional random fields, which are undirected graphical models that are ideally suited to handle complex, dependent observation vectors. For parameter learning from labeled training data we propose an algorithm based on averaged perceptron learning for hidden Markov models. Since our learning technique performs particle filtering as part of each optimization step, it is able to learn parameters that maximize filter performance, taking dependencies between measurements, sensor noise, and sample-based approximations into account.

We investigate the capabilities of CRF-Filters using the problem of laser-based robot localization. Experiments demonstrate that our approach is able to jointly learn the parameters of motion and sensor models, resulting in significant improvements over generatively trained particle filters. In fact, our approach learns different sensor models depending on the number of beams in a laser range-scan and on the localization task (global localization or tracking).

In this paper, we focused on standard motion and measurement models used in robot localization. However, we believe that CRF-Filters are applicable to a much wider range of sensors and state estimation problems. In future work we will investigate other sensor and motion models in the context of robot mapping and make use of the additional flexibility provided by the feature functions in CRF-Filters.

An important limitation of our current approach is the need for ground truth locations and we consider discriminative learning with partially labeled sequences a very promising direction for future work. Furthermore, more advanced learning algorithms such as L-BFGS need to be investigated in the context of CRF-Filters. Finally, Liao and colleagues [8] showed how to automatically extract good feature functions from high-dimensional, continuous measurements. We believe that the extension of this technique to our work would greatly increase the capabilities of CRF-Filters.

Acknowledgments

This work was supported by the NSF under CAREER grant number IIS-0093406, and by DARPA's ASSIST and CALO Programs (contract numbers: NBCH-C-05-0137, SRI subcontract 27-000968).

References

[1] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun. Discriminative training of Kalman filters. In *Proc. of Robotics: Science and Systems*, 2005.

[2] M. Bowling, D. Wilkinson, A. Ghodsi, and A. Milstein. Subjective localization with action respecting embedding. In *Proc. of the International Symposium of Robotics Research*, 2005.

[3] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2002.

[4] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, New York, 2001.

[5] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[6] K. Konolige, K. Chou. Markov localization using correlation. *Proc. of the International Joint Conference on Artificial Intelligence*, 1999.

[7] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning*, 2001.

[8] L. Liao, T. Choudhury, D. Fox, and H. Kautz. Training conditional random fields using virtual evidence boosting. In *Proc. of the International Joint Conference on Artificial Intelligence*, 2007.

[9] P. Pfaff, W. Burgard, and D. Fox. Robust Monte-Carlo Localization using adaptive likelihood models. In *Proc. of the European Robotics Symposium*, 2006.

[10] D. Stronger and P. Stone. Towards autonomous sensor and actuator model induction on a mobile robot. *Connection Sciences*, 18(2), 2006.

[11] E.B. Sudderth, A.B. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[12] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[13] L. Taycher, G. Shakhnarovich, D. Demirdjian, and T. Darrell. Conditional random people: Tracking humans with CRFs and grid filters. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[14] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, September 2005. ISBN 0-262-20162-3.

[15] S. Thrun, D. Fox, and W. Burgard. Monte Carlo localization with mixture proposal distributions. In *Proc. of the National Conference on Artificial Intelligence*, 2000.

[16] N. Vlassis, B. Terwijn, and B. Kröse. Auxiliary particle filter robot localization from high-dimensional sensor observations. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2002.