Discriminative Training of Kalman Filters

Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y. Ng and Sebastian Thrun
Department of Computer Science
Stanford University
Stanford, CA 94305

***Abstract*— Kalman filters are a workhorse of robotics and are routinely used in state-estimation problems. However, their performance critically depends on a large number of modeling parameters which can be very difficult to obtain, and are often set via significant manual tweaking and at a great cost of engineering time. In this paper, we propose a method for automatically learning the noise parameters of a Kalman filter. We also demonstrate on a commercial wheeled rover that our Kalman filter's learned noise covariance parameters—obtained quickly and fully automatically—significantly outperform an earlier, carefully and laboriously hand-designed one.**

I. Introduction

Over the past few decades, Kalman filters (KFs) [5] and extended Kalman filters (EKFs) [3] have found widespread applications throughout all branches of engineering. EKFs take as input sequences of measurements and controls, and output an estimate of the state of a dynamic system. They require a model of the system, comprised of a next state function, a measurement function, and the associated noise terms. EKFs are arguably one of the most influential Bayesian techniques in all of engineering and science.

This paper addresses a fundamental problem with the EKF: that of arriving at models suitable for accurate state estimation. The next state function and the measurement function are sometimes relatively easy to model, since they describe the underlying physics of the system. But even in applications where the next-state function and the measurement function are accurate, the *noise terms* are often difficult to estimate. The noise terms capture what the deterministic model fails to: the effects of unmodeled perturbations on the system.

The noise is usually the result of a number of different effects:

- Mis-modeled system and measurement dynamics.
- The existence of hidden state in the environment not modeled by the EKF.
- The discretization of time, which introduces additional error.
- The algorithmic approximations of the EKF itself, such as the Taylor approximation commonly used for linearization.

All these effects cause perturbations in the state transitions and measurements. In EKFs, they are commonly characterized as "noise." Further, the noise is assumed to be *independent* over time—whereas the phenomena described above cause highly correlated noise. The magnitude of the noise in an EKF is therefore extremely difficult to estimate. It is therefore surprising that the issue of *learning* noise terms remains largely unexplored in the literature. A notable exception is the filter tuning literature. (See, e.g., [7], [8] for an overview.) Although some of their ideas are fairly similar and could be automated, they focus mostly on a formal analysis of (optimally) reducing the order of the filter (for linear systems), and how to use the resulting insights for tuning the filter.

To further motivate the importance of optimizing the Kalman filter parameters (either by learning or tuning), consider the practical problem of estimating the variance parameter for a GPS unit that is being used to estimate the position x of a robot. A standard Kalman filter model would model the GPS readings x_{measured} as the true position plus noise:

$$x_{\text{measured}} \quad = \quad x_{\text{true}} + \varepsilon,$$

where ε is a noise term with zero mean and variance σ^2. The GPS' manufacturer specifications will sometimes explicitly give σ^2 for the unit; otherwise, one can also straightforwardly estimate σ^2 by placing the vehicle at a fixed, known, position, and measuring the variability of the GPS readings. However, in practice either of these choices for σ^2 will work very poorly if it is the parameter used in the Kalman filter. This is because GPS errors are often *correlated* over time, whereas the straightforward implementation of the Kalman filter assumes that the errors are independent. Thus, if the vehicle is stationary and we average n GPS readings, the filter assumes that the variance of the resulting estimate is σ^2/n. However, if the errors are correlated over time, then the true variance of the resulting position state estimate can be significantly larger than σ^2. The extreme of this case would be *full correlation*: If the errors were perfectly correlated so that all n readings are identical, then the variance of the average would be σ^2 instead of σ^2/n.[1] Thus, if σ^2 was the parameter used in the filter, it will tend to underestimate the *long time-scale* variance of the GPS readings, and perhaps as a result "trust" the GPS too much relative to other sensors (or relative to the dynamic model), and therefore give poor estimates of the state.

In practice, a number of authors have observed this effect on several robots including an autonomous helicopter platform using GPS for its state estimates [10], and ground rover platforms using a SICK LADAR to estimate their position [13]. In each of these cases, significant human time was expended to try to "tweak" the variance parameter to what they guessed

[1] More generally, we have that $\mathrm{Var}(\frac{1}{n}\sum_{i=1}^n x_i) = \frac{1}{n^2}\left(\sum_{i=1}^n \mathrm{Var}(x_i) + \sum_{i=1}^n \sum_{j=1,j\neq i}^n \mathrm{Cov}(x_i, x_j)\right)$.

Fig. 1. Ground rover platform instrumented with inexpensive IMU and GPS. This robot is commercially provided to contractors in the DARPA LAGR program.

were more "appropriate" values, so that the Kalman filter gives more accurate state estimates. For instance, in the GPS example above, by artificially choosing a larger σ^2 than that suggested by the manufacturer specifications, one would be able to reduce the KF's overconfidence in the GPS, and thereby obtain more accurate state estimates.

In this paper, we propose several machine learning algorithms for automatically choosing the noise parameters of the Kalman filter or extended Kalman filter (EKF). Our work is based on the fundamental assumption that during the EKF development, it is possible to instrument the system to measure additional variables, which provide highly accurate estimates for the state variables. These measurements are only available in the initial tuning phase; later they are no longer available.

An example of this setup is the robot shown in Figure 1. This figure shows a commercial robot equipped with a low-cost IMU (inertial measurement unit) and a low-end GPS receiver (global positioning system). Both are used to estimate the robot's geo-referenced location when the robot is in operation. The vendor of this unit supplies an EKF for pose estimation which has been optimized manually to achieve the best performance. A clear option—which the vendor did not exercise—would have been to attach a highly accurate differential GPS receiver to the unit to "tune" the EKF. With such a unit, it becomes possible to receive "ground truth" information on the actual coordinates of the robot.

This paper presents a family of learning algorithms that utilizes such information for learning the covariance parameters of an EKF. The idea is relatively straightforward: Train an EKF so that it maximizes its predictive accuracy, where "accuracy" is evaluated through the reference data obtained during training. The different algorithms provide different criteria for assessing the prediction accuracy of an EKF. Some simply measure the mean square error of the EKF; others measure the likelihood of the high-accuracy measurements. No matter what criterion is being used for training, however, the trained filters consistently outperform the EKF carefully tuned by hand. In fact, in our experiments we achieve results that are significantly more accurate than those provided by a commercial robot vendor. Thus, our approach promises to relieve EKF developers of the tedious task of tuning noise parameters by hand.

II. The Extended Kalman Filter

We will begin with a brief review of the EKF, defining the basic notation and terminology in this paper. Throughout this paper we use $z \sim P(z)$ to denote the random variable z has distribution P. We use $\mathcal{N}(z; \mu, \Sigma)$ to denote the formula for a density of a Gaussian with mean μ and covariance Σ, evaluated at z.

The EKF addresses the problem of state estimation in non-linear continuous dynamical systems. Here, we will formulate the EKF for the discrete time setting. At each point in time, the true state of the system being monitored will be denoted by x_t, where t is the time index.

The EKF assumes that state transitions are governed by a discrete-time continuous-state non-linear law of the form

$$x_t = f(x_{t-1}, u_t) + \varepsilon_t.$$

Here u_t is a control, f is a nonlinear function that maps the control and the state one time step earlier into the state at time t, and ε_t is the process noise. The process noise ε_t is Gaussian with zero mean and covariance R. Hence we have that $x_t \sim \mathcal{N}(x_t; f(x_{t-1}, u_t), R)$.

Measurements z_t are formed through a non-linear law of the form

$$z_t = g(x_t) + \delta_t.$$

Here z_t is a measurement, g is a nonlinear function of the state x_t, and δ_t is the measurement noise. The measurement noise is Gaussian with zero mean and covariance Q. Hence we have that $z_t \sim \mathcal{N}(z_t; g(x_t), Q)$.

The EKF then provides us with an estimate μ_t of the state at time t, along with an expected error of this estimate, expressed through a covariance Σ_t. Put differently, given a Gaussian estimate of the state specified by mean and covariance $\langle \mu_{t-1}, \Sigma_{t-1} \rangle$ at time $t-1$, the EKF update rule provides us with an estimate of both quantities at time t.

In detail, the update requires us to linearize the nonlinear function g and h through Taylor approximation. This is usually written as follows:

$$\begin{aligned} f(x_{t-1}, u_t) &\approx f(\mu_{t-1}, u_t) + F_t\,(x_{t-1} - \mu_{t-1}), \\ g(x_t) &\approx g(\mu_t) + G_t\,(x_t - \mu_t). \end{aligned}$$

Here F_t and G_t are Jacobian matrices of f and g, respectively, taken at the filter estimate μ.

The resulting state transition and measurement functions are now *linear* in x. For linear systems, the Kalman filter produces an exact update, by manipulating the various Gaussians involved. The update is then usually factored into two separate steps, a prediction step, and a measurement update step. The prediction step starts with the estimate μ_{t-1} and its covariance

Σ_{t-1} at time $t-1$, and produces an estimate for time t:

$$\begin{aligned}\bar{\mu}_t &= f(\mu_{t-1}, u_t),\\ \bar{\Sigma}_t &= F_t\, \Sigma_{t-1}\, F_t^\top + R.\end{aligned}$$

The bar in $\bar{\mu}_t$ and $\bar{\Sigma}_t$ indicates that these estimates are pure predictions, before taking the measurement z_t into account. This happens in the measurement update step, in which the EKF integrates the measurement z_t by first calculating the Kalman gain:

$$K_t = \bar{\Sigma}_t\, G_t^\top\, (G_t\, \bar{\Sigma}_t\, G_t^\top + Q)^{-1}.$$

This expression specifies the amount by which the estimate will be adjusted in accordance to the measurement prediction error $z_t - g(\mu_t)$. This leads to the update of the mean and variance

$$\begin{aligned}\mu_t &= \bar{\mu}_t + K_t\, (z_t - g(\bar{\mu}_t)),\\ \Sigma_t &= (I - K_t\, G_t)\, \bar{\Sigma}_t.\end{aligned}$$

To implement an EKF, the designer needs to determine two sets of things: The nonlinear functions g and f, and the noise covariances R and Q. While f and g can sometimes be obtained through basic physics considerations,[2] the covariances R and Q are difficult to estimate. It is common practice to manually tune those matrices until the EKF exhibits the desired performance. In the presence of ground truth data, one could try to tune the parameters such that the filter estimates are as accurate as possible in estimating the ground truth data. Manual tuning with such an objective is effectively manual discriminative training of the Kalman filter parameters. In the next section we present learning procedures that automate such a tuning process.

III. Learning the filter parameters

We now describe our learning techniques for obtaining the noise parameters of the Kalman filter automatically. For simplicity, our discussion will focus on learning R and Q, though all the presented methods also apply more generally. All but one of our approaches requires that one is given a highly accurate instrument for measuring either all or a subset of the variables in the state x_t. Put differently, in the EKF learning phase, we are given additional values $y_1, y_2, \ldots$, where each y_t is governed by a projective equation of the type

$$y_t = h(x_t) + \gamma_t.$$

Here h is a function, and γ_t is the noise with covariance P. In our example below, y_t are the readings from a high-end GPS receiver. The function h will be a projection which extracts the subset of the variables in x_t that correspond to the Cartesian coordinates of the robot.

[2]One common exception are the "damping" term in the state dynamics. For example, if we estimate the gyros of an IMU, or indeed any other sensor, to have a slowly varying bias (as is commonly done in practice), the bias is usually modeled as $x_t = \lambda x_{t-1} + \epsilon_t$, where $0 < \lambda < 1$ governs the rate at which the bias x_t tends towards zero. The parameters λ and $\mathrm{Var}(\epsilon_t)$ jointly govern the dynamics of the bias, and λ is an example of a parameter in the state update equation that is difficult to estimate and is, in practice, usually tuned by hand.

Let $x_{0:T}$ denote the entire state sequence $(x_0, x_1, \ldots, x_T)$, and similarly let $u_{1:T}$, $y_{0:T}$ and $z_{0:T}$ denote the corresponding observation sequences. Assuming that we have a prior $p(x_0)$ on the initial state at time 0, the state update equation and the observation equation (together with the known controls $u_{1:T}$) define a joint probability distribution on $x_{0:T}$, $y_{0:T}$, and $z_{0:T}$. Specifically,

$$\begin{aligned}&p(x_{0:T}, y_{0:T}, z_{0:T} \mid u_{1:t}) \qquad (1)\\ &= p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1}, u_t) \prod_{t=0}^{T} p(y_t|x_t)\, p(z_t|x_t),\end{aligned}$$

where

$$\begin{aligned}p(x_t|x_{t-1}, u_t) &= \mathcal{N}(x_t; f(x_{t-1}, u_t), R), && (2)\\ p(y_t|x_t) &= \mathcal{N}(y_t; h(x_t), P), && (3)\\ p(z_t|x_t) &= \mathcal{N}(z_t; g(x_t), Q). && (4)\end{aligned}$$

Using the linearization approximations to f and g (h, being a projection operation, is assumed to be linear), the joint distribution $p(x_{0:T}, y_{0:T}, z_{0:T})$ defined by the EKF model is actually a joint linear-Gaussian model. [12] Since the joint distribution is well-defined, so are other quantities such as the marginal distributions (such as $p(y_{0:T}, z_{0:T}) = \int_{x_{0:T}} p(x_{0:T}, y_{0:T}, z_{0:T}) dx_{0:T}$) and the conditional distributions (such as $p(z_{0:T}|y_{0:T})$) over these same random variables.

A. Generative Approach: Maximizing The Joint Likelihood

We will first discuss a naive approach, which requires access to the *full* state vector. Put differently, this approach requires that h is the identity function, and that the noise in γ is so small that it can safely be neglected. While this approach is generally inapplicable simply because it is often difficult to measure *all* state variables, it will help us in setting up the other approaches.

Generative learning proceeds by maximizing the likelihood of all the data. Since in this section we assume the full state vector is observed (i.e., for all t we have $y_t = x_t$), the covariance matrices $\langle R_{\mathrm{joint}}, Q_{\mathrm{joint}} \rangle$ are estimated as follows:

$$\langle R_{\mathrm{joint}}, Q_{\mathrm{joint}} \rangle = \arg\max_{R,Q} \log p(x_{0:T}, z_{0:T}|u_{1:T}). \qquad (5)$$

Now by substituting in Eqn. (1), (2) and (4) into Eqn. (5) we get that the optimization decomposes and we can estimate R_{joint} and Q_{joint} independently as:

$$\begin{aligned}R_{\mathrm{joint}} &= \arg\max_R -T \log|2\pi R|\\ &\quad - \sum_{t=1}^{T} (x_t - f(x_{t-1}, u_t))^\top R^{-1} (x_t - f(x_{t-1}, u_t)),\\ Q_{\mathrm{joint}} &= \arg\max_Q -(T+1) \log|2\pi Q|\\ &\quad - \sum_{t=0}^{T} (z_t - g(x_t))^\top Q^{-1} (x_t - g(x_t)).\end{aligned}$$

An interesting observation here is that both the objective for R_{joint} and the objective for Q_{joint} decompose into two terms: a term from the normalizer whose objective is to deflate the determinant of R (Q), and one that seeks to minimize a quadratic function in which the inverse of R (Q) is a factor, and which therefore seeks to inflate R (Q). The optimal R_{joint} and Q_{joint} can actually be computed in closed form and are given by

$$\begin{aligned} R_{\text{joint}} &= \frac{1}{T}\sum_{t=1}^{T}(x_t - f(x_{t-1}, u_t))(x_t - f(x_{t-1}, u_t))^\top, \\ Q_{\text{joint}} &= \frac{1}{T+1}\sum_{t=0}^{T}(z_t - g(x_t))(x_t - g(x_t))^\top. \end{aligned}$$

Note the naive approach never actually *executes* the filter for training. It simply trains the elements of the filter. It therefore implicitly assumes that training the elements individually is as good as training the EKF as a whole.

B. Minimizing The Residual Prediction Error

The technique of maximizing the joint likelihood, as stated above, is only applicable when the full state is available during training. This is usually not the case. Often, h is a function that projects the full state into a lower-dimensional projection of the state. For example, for the inertial navigation system described below, the full state involves bias terms of a gyroscope that cannot be directly measured. Further, the technique of minimizing the conditional likelihood never actually *runs* the filter! This is a problem if noise is actually correlated, as explained in the introduction of this paper.

A better approach, thus, would involve training an EKF that minimizes the prediction error for the values of y_t. More specifically, consider the EKF's prediction of y_t:

$$E[y_t \mid u_{1:t}, z_{0:t}] = h(\mu_t). \tag{6}$$

Here, μ_t is the result of running the EKF algorithm (with some variance parameters R and Q for the filter), and taking its estimate for the state x_t after the EKF has seen the observations $z_{0:t}$ (and the controls $u_{1:t}$). Therefore μ_t depends implicitly on R and Q.

The prediction error minimization technique simply seeks the parameters R and Q that minimize the quadratic deviation of y_t and the expectation above, weighted by the inverse covariance P:

$$\langle R_{\text{res}}, Q_{\text{res}} \rangle = \arg\min_{R,Q} \sum_{t=0}^{T} (y_t - h(\mu_t))^\top P^{-1}(y_t - h(\mu_t)).$$

If P is any multiple of the identity matrix, this simplifies to

$$\langle R_{\text{res}}, Q_{\text{res}} \rangle = \arg\min_{R,Q} \sum_{t=0}^{T} ||y_t - h(\mu_t)||_2^2. \tag{7}$$

Thus, we are simply choosing the parameters R and Q that cause the filter to output the state estimates that minimize the squared differences to the measured values y_t.

This optimization is more difficult than maximizing the joint likelihood. The error function is not a simple function of the covariances R and Q. Instead, it is being mediated through the mean estimates μ_t, which depend on the covariances R and Q in a complicated way. The mean estimates μ_t are the result of running an EKF over the data. Hence, this learning criterion evaluates the actual performance of the EKF, instead of its individual components.

Computing the gradients for optimizing the residual prediction error is more involved than in the previous case. However, an optimization that does not require explicit gradient computations, such as the Nelder-Mead simplex algorithm, can also be applied. [11]

C. Maximizing The Prediction Likelihood

The objective in Eqn. (7) measures the quality of the state estimates μ_t output by the EKF, but does not measure the EKF's estimates of the *uncertainty* of its output. At each time step, the EKF estimates both μ_t and a covariance for its error Σ_t. In applications where we require that the EKF gives accurate estimates of its uncertainty [15], we choose instead the prediction likelihood objective

$$\langle R_{\text{pred}}, Q_{\text{pred}} \rangle = \arg\max_{R,Q} \sum_{t=0}^{T} \log p(y_t \mid z_{0:t}, u_{1:t}). \tag{8}$$

Here the y_t's are treated as measurements. This training regime trains the EKF so as to maximize the probability of these measurements.

The probability $p(y_t \mid z_{1:t}, u_{1:t})$ can be decomposed into variables known from the filter:

$$p(y_t \mid z_{0:t}, u_{1:t}) = \int p(y_t \mid x_t) \underbrace{p(x_t \mid z_{0:t}, u_{1:t})}_{\sim \mathcal{N}(x_t; \mu_t, \Sigma_t)} dx_t.$$

Under the Taylor expansion, this resolves to

$$p(y_t \mid z_{0:t}, u_{1:t}) = \mathcal{N}(y_t; h(\mu_t), H_t \, \Sigma_t \, H_t^\top + P). \tag{9}$$

Here H_t is the Jacobian of the function h. The resulting maximization of the log likelihood gives us

$$\begin{aligned} \langle R_{\text{pred}}, Q_{\text{pred}} \rangle = \arg\max_{R,Q} \sum_{t=0}^{T} & -\log|2\pi\Omega_t| \\ & -(y_t - h(\mu_t))^\top \, \Omega_t^{-1}(y_t - h(\mu_t)). \end{aligned}$$

Here we abbreviated $\Omega = H_t \; \Sigma_t \; H_t^\top + P$. Once again, this optimization involves the estimate μ_t, through which the effects of P and Q are mediated. It also involves the covariance Σ_t. We note when the covariance P is small, we can omit it in this expression.

This objective should also be contrasted with Eqn. (7). The difference is that here the filter is additionally required to give "confidence rated" predictions by choosing covariances Σ_t that reflect the true variability of its state estimates μ_t.

D. Maximizing The Measurement Likelihood

We now apply the idea in the previous step to the measurement data $z_{0:T}$. It differs in the basic assumption: Here we do not have additional data $y_{1:T}$, but instead have to tune the EKF simply based on the measurements $z_{0:T}$ and the controls $u_{1:T}$.

Recalling that the EKF model, for fixed $u_{1:t}$, gives a well-defined definition for the joint $p(x_{0:t}, z_{0:t} \mid u_{1:t})$, the marginal distribution $p(z_{0:t} \mid u_{1:t})$ is also well defined. Thus, our approach is simply to choose the parameters that maximize the likelihood of the observations in the training data:

$$\langle R_{\mathrm{meas}}, Q_{\mathrm{meas}}\rangle = \arg\min_{R,Q} \log p(z_{0:T} \mid u_{1:T}).$$

The value of the objective is easily computed by noting that, by the chain rule of probability,

$$p(z_{0:T}|u_{1:T}) = \prod_{t=0}^{T} p(z_t|z_{0:t-1}, u_{1:T}).$$

Moreover, each of the terms in the product is given by

$$\begin{aligned} & p(z_t|z_{0:t-1}, u_{1:T}) \\ &= \int_{x_t} p(z_t|x_t, z_{0:t-1}, u_{1:T}) p(x_t|z_{0:t-1}, u_{1:T}) dx_t \\ &= \int_{x_t} p(z_t|x_t) p(x_t|z_{0:t-1}, u_{1:t}) dx_t. \end{aligned}$$

The term $p(z_t|x_t)$ is given by Eqn. (4), and $p(x_t|z_{1:t-1}, u_{1:t-1}) = \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t)$, where $\bar{\mu}_t$ and $\bar{\Sigma}_t$ are quantities computed by the EKF. Thus this approach also runs the EKF to evaluate its performance criterion. However since no ground truth data is used here, the performance criterion is not predictive performance for the state sequence (which is what we ultimately care about), but merely predictive performance on the observations $z_{0:T}$.

E. Optimizing The Performance After Smoothing

The two discriminative criteria of Sections III-B and III-C evaluate the performance of the covariance matrices $\langle R, Q\rangle$ as used in the EKF. These criteria can easily be extended to the smoothing setting. (See, e.g., [8] for details on smoothing.) In particular let $\tilde{\mu}_t$ be the state estimates as obtained from the smoother, then the smoother equivalent of Eqn. (7) is:

$$\langle R_{\mathrm{res-sm}}, Q_{\mathrm{res-sm}}\rangle = \arg\min_{R,Q} \sum_{t=0}^{T} \|y_t - h(\tilde{\mu}_t)\|_2^2.$$

The smoother likelihood objective is given by conditioning on *all* observations (instead of only up to time t as in the filter case). So the smoother equivalent of Eqn. (8) is:

$$\langle R_{\mathrm{pred-sm}}, Q_{\mathrm{pred-sm}}\rangle = \arg\max_{R,Q} \sum_{t=0}^{T} \log p(y_t|z_{0:T}, u_{1:T}).$$

The smoother likelihood objective is closely related to the training criteria used for conditional random fields which are widely used in machine learning to predict a sequence of labels (states) from *all* observations. (See, e.g., [6] and [4] for details.)

The two criteria proposed in this section are optimizing the covariance matrices $\langle R, Q\rangle$ for smoother performance, not filter performance. So we expect the resulting covariance matrices $\langle R, Q\rangle$ (although good for smoothing) not to be optimal for use in the filter. This is confirmed in our experiments.

F. Training

The previous text established a number of criteria for training covariance matrices; in fact, the criteria make it possible to also tune the functions f and g, but we found this to be of lesser importance in our work.

The training algorithm used in all our experiments is a coordinate ascent algorithm: Given initial estimates of R and Q, the algorithm repeatedly cycles through each of the entries of R and Q. For each entry p, the objective is evaluated when decreasing and increasing the entry by α_p percent. If the change results in a better objective, the change is accepted and the parameter α_p is increased by ten percent, otherwise α_p is decreased by fifty percent. Initially we have $\alpha_p = 10$. We find empirically that this algorithm converges reliably within 20-50 iterations.

IV. EXPERIMENTS

We carried out experiments on the robot shown in Figure 1. This is a differential drive robot designed for off-road navigation. For state estimation, it is instrumented with a low cost GPS unit; a low cost inertial measurement unit (IMU) consisting of 3 accelerometers for measuring linear accelerations, and 3 gyroscopes for measuring rotational velocities; a magnetometer (magnetic compass); and optical wheel encoders (to measure forward velocity, assuming rigid contact with the ground). The GPS is WAAS enabled, and returns position estimates at 1Hz with a typical position accuracy of about 3 meters.

These vehicles were built by Carnegie Mellon University for a competition in which each team obtains an identical copy of the vehicle, which can be used for software development. The software developed by each team will then be tested on a separate (but identical) vehicle at a Carnegie Mellon site. Since we have our own vehicle, we are able to install an accurate GPS unit onto it to get additional, more accurate, state estimates during development time. Specifically, we mounted onto our vehicle a Novatel RT2 differential GPS unit, which gives position estimates y_t at 20Hz to about 2cm accuracy. While we could use the accurate GPS unit for development, the hardware on which our algorithms will be evaluated will not have the more accurate GPS.

The vehicle also comes with a carefully hand-tuned EKF. Since this pre-existing EKF was built by a highly experienced team of roboticists at Carnegie Mellon (not affiliated with the authors), we believe that it represents an approximate upper-bound on the performance that can reasonably be expected in a system built by hand-tweaking parameters (without using ground truth data). We therefore evaluate our learning algorithms against this hand-designed EKF.

The state of the vehicle is represented as a five dimensional vector, including its map coordinates x_t and y_t, orientation θ_t, forward velocity v_t and heading gyro bias b_t. [2] The measurement error of a gyroscope is commonly characterized as having a Gaussian random component, and an additive bias term that varies slowly over time. Neglecting to model the bias of the gyroscope will lead to correlated error in the robot's heading over time, which will result in poor estimation performance.

More formally, our EKF's state update equations are given by:

$$\begin{aligned}
x_t &= x_{t-1} + \Delta t\ v_{t-1} \cos\theta_{t-1} + \ \varepsilon_t^{\text{for}} \cos\theta_{t-1} \\
&\quad - \varepsilon_t^{\text{lat}} \sin\theta_{t-1}, \\
y_t &= y_{t-1} + \Delta t\ v_{t-1} \sin\theta_{t-1} + \ \varepsilon_t^{\text{for}} \sin\theta_{t-1} \\
&\quad + \varepsilon_t^{\text{lat}} \cos\theta_{t-1}, \\
\theta_t &= \theta_{t-1} + \Delta t\ (r_t + b_t) + \varepsilon_t^{\theta}, \\
v_t &= v_{t-1} + \Delta t\ a_t + \varepsilon_t^{v}, \\
b_t &= b_{t-1} + \varepsilon_t^{b}.
\end{aligned}$$

Here $\varepsilon_t^{\text{for}}$ and $\varepsilon_t^{\text{lat}}$ are the position noise in the forward and lateral direction with respect to the vehicle. The control is $u_t = (r_t \ \ a_t)^\top$, where r_t is the rotational velocity command, and a_t is the forward acceleration.

The observation equations are given by

$$\begin{aligned}
\tilde{x}_t &= x_t + \delta_t^x, \\
\tilde{y}_t &= y_t + \delta_t^y, \\
\tilde{\theta}_t &= \theta_t + \delta_t^\theta, \\
\tilde{v}_t &= v_t + \delta_t^v.
\end{aligned}$$

In our model, ε_t is a zero mean Gaussian noise variable with covariance $\text{diag}(\sigma^{\text{for}}, \sigma^{\text{lat}}, \sigma^\theta, \sigma^v, \sigma^b)$. Similarly, δ_t is a zero mean Gaussian noise variable with covariance $\text{diag}(\gamma^x, \gamma^y, \gamma^\theta, \gamma^v)$. In our experiments, the nine parameters $\sigma^{\text{for}}, \sigma^{\text{lat}}, \sigma^\theta, \sigma^v, \gamma^x, \gamma^y, \gamma^\theta, \gamma^v$ were fit using the learning algorithms. Furthermore, our model assumed that $\gamma^x = \gamma^y$.

Our experimental protocol was as follows. We collected two sets of data (100s each) of driving the vehicle around a grass field, and used one for training, the other for testing. Because the observations y_t do not contain the complete state (but only position coordinates), the "naive approach" of maximizing the joint likelihood is not directly applicable. However the highly accurate position estimates allow us to extract reasonably accurate estimates of the other state variables.[3] Using these state estimates as a substitute for the real states, we estimate the covariances $\langle R_{\text{joint}}, Q_{\text{joint}} \rangle$ using the joint likelihood criterion. The estimates $\langle R_{\text{joint}}, Q_{\text{joint}} \rangle$ are used for initialization when using the other criteria (which do not have closed form solutions).

We evaluate our algorithms on test data using two error metrics. The first is the RMS error in the estimate of the vehicle's position (cf. Eqn. 7):

$$\left(\frac{1}{T} \sum_{t=1}^{T} ||h(\mu_t) - y_t||^2 \right)^{1/2}.$$

Above, μ_t is the EKF estimate of the full state at time t, and $h(\mu_t)$ is the EKF estimate of the 2D coordinates of the vehicle at time t.

The second error metric is the prediction log-loss

$$-\frac{1}{T} \sum_{t=1}^{T} \log p(y_t \mid z_{0:t}, u_{1:t}).$$

Following the discussion in Section III.C, the main difference between these two metrics is in whether it demands that the EKF gives accurate covariance estimates.

The highly accurate GPS outputs position measurements at 20Hz. This is also the frequency at which the built-in hand-tuned filter outputs its state estimates. We use the corresponding time discretization $\Delta t = .05s$ for our filter.

Each of our learning algorithms took about 20-30 minutes to converge. Our results are as follows (smaller values are better):[4]

Learning Algorithm	RMS error	log-loss
Joint	0.2866	23.5834
Res	0.2704	1.0647
Pred	0.2940	-0.1671
Meas	0.2943	60.2660
Res-sm	0.3229	2.9895
Pred-sm	0.5831	0.4793
CMU hand-tuned	0.3901	0.7500

In this table, "Res" stands for the algorithm minimizing the residual prediction error ($\langle R_{\text{res}}, Q_{\text{res}} \rangle$), etc.

As expected the filters learned using the smoother criteria of Section III-E (Res-sm,Pred-sm) are outperformed by the filters learned using the corresponding filter criteria (Res,Pred). So from here on, we will not consider the filters learned using the smoother criteria.

We see that the hand-tuned EKF had an RMS error of about 40cm in estimating the position of the vehicle, and that all of our learned filters obtain significantly better performance. Using the parameters learned by maximizing the prediction likelihood ($\langle R_{\text{pred}}, Q_{\text{pred}} \rangle$), we also obtain better log-loss (negative log likelihood). Minimizing the residual prediction error on the training data results in smallest residual error on the test data. Similarly, minimizing the log-loss (or, equivalently, maximizing the prediction likelihood) on the training data results in smallest log-loss on the test data. Thus, discriminative training allows us to successfully optimize for the criteria we care about. We also notice that, although the filters trained by joint likelihood maximization and measurement likelihood maximization have small RMS error, they perform poorly on the log-loss criterion. This can be explained by

[3]More specifically, we ran an extended Kalman smoother to obtain estimates for θ_t, v_t, b_t. This smoother used very high variances for the measured $\tilde{\theta}_t, \tilde{v}_t$, and very small variances for the position measurements $\tilde{x}_t, \tilde{y}_t$. The smoother also assumed very high process noise variances, except for the gyro bias term. This choice of variances ensures the smoother extracts state estimates that are consistent with the highly accurately measured position coordinates. The results from the smoother were not very sensitive to the exact choice of the variances. In the reported experiments, we used $\text{diag}(1, 1, 1, 1, .001^2)$ for the process noise and $\text{diag}(.02^2, .02^2, 10^2, 10^2)$ for the measurement noise.

[4]All results reported are averaged over two trials, in which half of data is used for training, and the other half for testing.

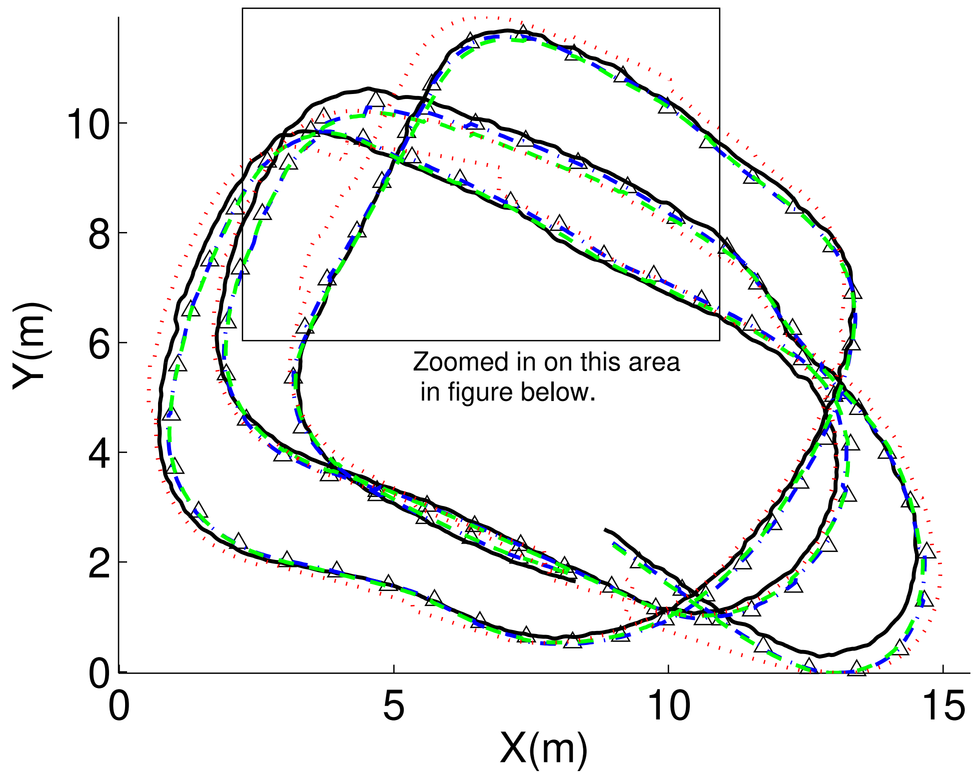

Fig. 2. Typical state estimation results. Plot shows ground truth trajectory (black solid line); on-board (inexpensive) GPS measurements (black triangles); estimated state using the filter learned by minimizing residual prediction error (blue dash-dotted line); estimated state using the filter learned maximizing the prediction likelihood (green dashed line); and estimated state using the CMU hand-tuned filter (red dotted line). (Colors where available.)

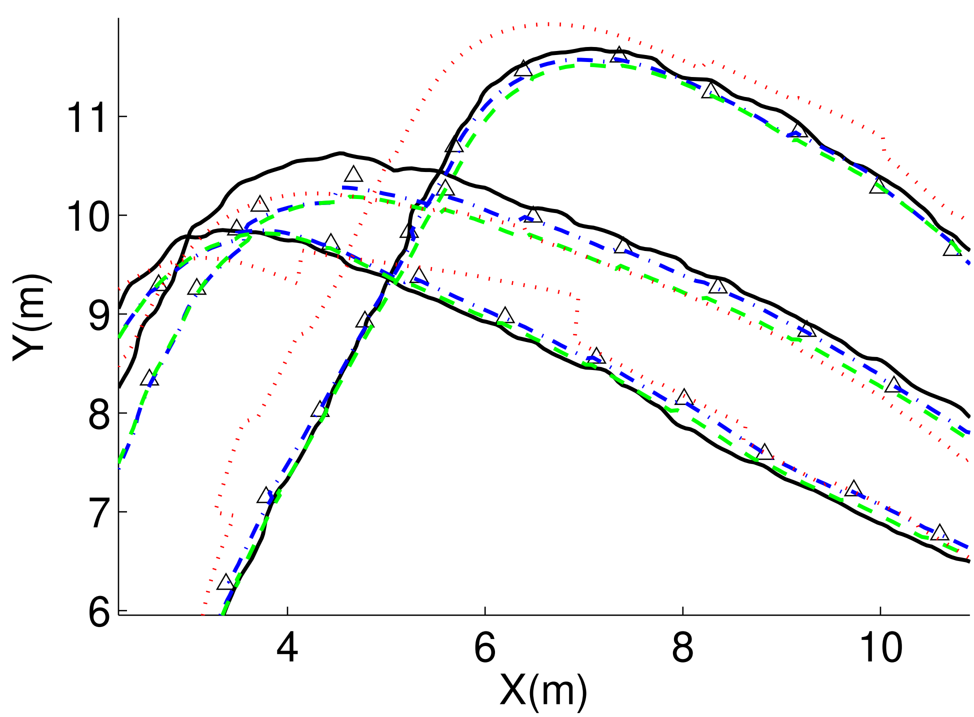

Fig. 3. Close-up of part of Figure 2. (Same legend as previous figure.)

correlated noise. More specifically, correlated noise causes the model trained by maximizing the joint likelihood to be overconfident about its state estimates, which results in the larger log-loss. The effect of correlated noise on the model trained by maximizing the measurement likelihood is even more significant. The model learns very small measurement variances, which allow it to track the measurements more closely. Unfortunately, in the case of correlated noise, tracking the measurements more closely does not mean tracking the state of the system more closely. The small measurement variances result in significant overconfidence in the state estimates, which causes the log-loss to be significantly higher.

Figure 2 shows a typical trajectory taken by the vehicle, as well as the estimates from two of the learned filters and the CMU hand-tuned filter. It is visually fairly clear from the figure that the learned filters are more accurately tracking the ground truth trajectory than the hand-tuned filter. Figure 3 shows a close-up of part of Figure 2. To reduce clutter, we have plotted only the output of two of the six learned filters here; however all learned filters have outputs that look visually very similar. One exception is that—as explained in the previous paragraph—filters learned by maximizing the measurement likelihood tend to follow the (often noisy) GPS measurements more closely.

V. Conclusion

We have presented a highly related family of algorithms for training the noise parameters of an EKF. All algorithms follow the same idea: Adjust the covariances of the EKF in a way that maximizes their predictive accuracy. Experimental results show that this training routine has two major advantages over previous best practice. First, it eliminates the need for a hand-tuning phase, thereby making it easier to develop working EKFs. Second, we find that the learned EKF is more accurate than even a well-tuned EKF constructed by hand. Among the learned filters, the best results were obtained by using discriminative training, which evaluates candidate covariances by evaluating the predictive performance of the EKF when using these covariances.

In our experiments we compare to a commercial EKF, provided as part of a robot developed for a major DARPA program. Clearly, it is difficult to assess how much tuning went into the EKF, and whether this is actually as good as can be done through manual tweaking. However, the EKF is central to the application of this robotic system, and we expect the development team spent at least a few days developing this EKF. Our approach outperforms this EKF by a large margin, based on a few minutes of data and a few minutes of learning. This suggests that our approach may yield better results with less development time.

We note that our training approach is also applicable to broader problems of EKF training. In particular, we chose not to learn the physical model as expressed in the state transition and the measurement functions. However, given an appropriate parameterization of these functions, it appears to be feasible to tune those functions as well. However, it remains an open question to which extent over-fitting poses a problem when doing so in practice.

The holistic training algorithms presented in this paper are highly related to an ongoing debate in the field of machine learning on using *discriminative* vs. *generative* algorithms for supervised learning. There, the consensus (assuming there is ample training data) seems to be that it is usually better to directly minimize the loss with respect to the ultimate performance measure, rather than an intermediate loss function such as the likelihood of the training data; see, e.g., [14], [9], [1]. This is because the model—no matter how complicated—is almost always not completely "correct" for the problem data. By analogy, when choosing the noise parameters for an EKF, we are interested in choosing parameters that lead to the EKF outputting accurate state estimates, rather than necessarily choosing the noise parameters that most correctly reflects each measurement's true variance (such as would be obtained from the maximum likelihood estimate or from most manufacturer specs, as discussed above).

Acknowledgments

We give warm thanks to Andrew Lookingbill and David Lieb for collecting the data from the LAGR robot platform. This work was supported by the DARPA LAGR program under contract number FA8650-04-C-7134.

References

[1] Pieter Abbeel and Andrew Y. Ng. Learning first order Markov models for control. In *NIPS 17*, 2005.

[2] J. A. Farrell and M. Barth. *The Global Positioning System and Inertial Navigation*. McGraw Hill, 1998.

[3] Arthur Gelb, editor. *Applied Optimal Estimation*. MIT Press, 1974.

[4] Sham Kakade, Yee Whye Teh, and Sam Roweis. An alternative objective function for Markovian fields. In *Proc. ICML*, 2002.

[5] Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[6] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

[7] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, 1982.

[8] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 2. Academic Press, 1982.

[9] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS 14*, 2002.

[10] A.Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Singapore, 2004. Springer Tracts in Advanced Robotics (STAR).

[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[12] Sam Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. Technical report, University of Toronto, 6 King's College Road, Toronto M5S 3H5, Canada, 1997.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[14] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[15] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proc. ICML*, 2001.