Kalman Filters for Prediction and Tracking in an Adaptive Sensor Network

Victoria Manfredi, Sridhar Mahadevan, Jim Kurose*

Technical Report 2005-7
February 24, 2005

Department of Computer Science
140 Governors Drive
University of Massachusetts
Amherst, Massachusetts 01003-4601
{vmanfred, mahadeva, kurose}@cs.umass.edu

Abstract

We consider the problem of configuring sensors in an adaptive sensor network being used to monitor meteorological features. One way to decide future sensor configurations is to base them on information currently being collected. For instance, if the network is being used to monitor storms in Oklahoma, then the sensors could be dynamically configured based on the predicted storm locations. While Kalman filters and their extensions are commonly used for prediction and tracking, they have been primarily applied to objects with known or fixed dynamics such as missiles or people. We explore the advantages and limitations of using Kalman filters to track objects with nonstationary dynamics (e.g., a storm can grow in size). In particular, we focus on tracking meteorological features over time with the objective of using this information to intelligently configure radars. We present results for tracking storm cells comparing least-squares regression with Kalman filter and switching Kalman filter methods.

Keywords: sensor networks, Kalman filters, storm tracking

1 Introduction

A sensor network is comprised of sensors collecting data in an environment. An adaptive sensor network is a sensor network in which the sensor configurations can be changed, either by users or under system control. Changing a sensor configuration may involve sensors moving autonomously from one location to another. Alternatively, it may involve changing how sensors sense their environment, such as by changing the frequencies at which measurements are taken, or by changing the locations where *in situ* or remotely sensed measurements are taken. The ability to dynamically change sensor configurations provides a great deal of flexibility in how, when, where, and what information is collected. The utility of this adaptability is

*This work was supported in part by the National Science Foundation under grants EEC-0313747001 and ECS-0218125. Any opinons, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Victoria Manfredi was supported by a National Science Foundation graduate research fellowship.

most apparent when the sensor network's resources are limited, such as when all locations cannot be sensed all of the time.

One approach for determining future sensor configurations is to base them on information currently being collected or that has been collected in the past. For instance, suppose an adaptive sensor network is being used to monitor storms in Oklahoma. The future paths of the storms being monitored can be predicted based on current and previous monitored storm data. Then if a storm is predicted at a certain latitude and longitude in five minutes, the radars can be configured to scan that location at that time.

In this paper we examine the problem of tracking storm cells over time with the objective of using current and past observations to intelligently configure radars. The context for this research is the NSF-funded CASA project (Collaborative Adaptive Sensing of the Atmosphere). Although the CASA distributed radar network will monitor a variety of meteorological features, including tornadoes, storm cells and shear, this paper focuses only on tracking storm cells. We note that our focus on tracking only storm cells is due to a lack of track data for tornadoes and other meteorological features. Tornado tracking and prediction would be very interesting and we hope that this work can be extended to such problems.

This paper examines the use of the Kalman filter and the switching Kalman filter for tracking storm cells. There are several advantages to using Kalman filters. First, prediction using the basic Kalman filter is extremely fast and requires little memory. This is essential for the real-time requirements of the CASA system. Second, an error estimate is associated with each prediction. Third, these predictions can be computed recursively, bounding the time and memory needed for computation. While the Kalman filter and its extensions are commonly used for prediction and tracking, they have been primarily applied to objects with known or fixed dynamics such as missiles or people. In order to use Kalman filters to track storms (or other meteorological features), several potential problems must be addressed. First, storms can change in size and intensity; they can also split into separate storms, or merge with other storms. Second, different storms do not necessarily have the same dynamics and an individual storm's dynamics will likely not satisfy the linear-Gaussian assumption of the Kalman filter. Thus, one of the goals of this paper is to examine the trade-offs of using the Kalman filter for tracking meteorological features.

The rest of this paper is organized as follows. Section 2 gives background on the prediction methods we use. Section 3 reviews current tracking methods. Section 4 formulates the problem we are addressing and describes our Kalman filter-based approach. Section 5 presents experimental results for storm cell tracking comparing least-squares regression, Kalman filter, and switching Kalman filter methods. Finally, Section 6 summarizes important issues identified in this project and Section 7 outlines future work.

2 Background

This section provides a brief review of least-squares regression, describes the graphical models framework, and explains the basics of Kalman filters, switching Kalman filters, and inference.

2.1 Linear Least-Squares Regression

See [23] for a more comprehensive treatment of least-squares regression. Given a set of points, linear regression determines the coefficients, slope and y-intercept, of the best fit line. This can be formulated as solving the equation $A\mathbf{x} = \mathbf{b}$ for the coefficent vector $\mathbf{x}$. Each row of the matrix A is associated with a data point. The first column of A is a vector of ones while the second column is a vector of values for the predictor variable. The variable $\mathbf{b}$ is a vector of values for the response variable.

If there are more than two points, there will be more equations than unknowns, overconstraining the problem and making it difficult to determine exact coefficients, if they exist. In this case, the best fit line is defined to be the one that minimizes the squared vertical distance (or error) from each point to the line. The error $A\mathbf{x} - \mathbf{b}$ is the difference between what the model $\mathbf{x}$ says the response value should be for a given

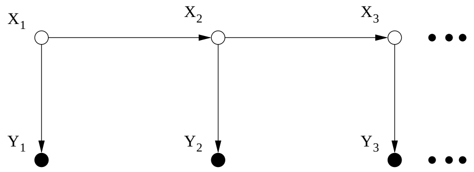


Figure 1: DBN formulation of the Kalman filter. X represents state and Y represents observations. Clear nodes are hidden and shaded nodes are observed.

predictor value, and what it actually is. The squared error is therefore given by $||A\mathbf{x} - \mathbf{b}||^2$. Mathematically, minimizing the squared error reduces to solving the following equation for the coefficient vector $\mathbf{x}$,

$$A^T A\mathbf{x} = A^T\mathbf{b}$$

2.2 Graphical Models

See [16, 22] for a more comprehensive treatment of graphical models. A graphical model efficiently encodes a joint probability distribution using a graph. Nodes in the graph represent random variables; edges in the graph capture dependencies between variables. Based on the dependencies being encoded, edges may be directed, as in a Bayesian network (BN), or undirected as in a Markov field. Each node that represents a discrete random variable has an associated conditional probability table (CPT). Each node that represents a continuous random variable has an associated conditional probability distribution (CPD). For all possible node and parent values, the CPT or CPD for a given node contains the probability, either in the form of a look-up table or distribution, that the node has some value conditioned on a set of values for its parents.

The Kalman filter and switching Kalman filter can be represented by models in one class of directed graphical models, specifically dynamic Bayesian networks (DBN). In a DBN, the values of the variables change over time according to a (probabilistic) transition model; while the values of the variables change, the transition model does not. A DBN can be thought of as a BN repeated for multiple time steps, with additional edges to connect variables at time t with variables at time $t+1$. In both BNs and DBNs, the problem of inference is to determine the probability distribution for a query variable (or variables) given some evidence. For instance, a DBN with variables x and y could be queried for the probability that variable x takes on some value at time t given observed values of y for time steps 1 to t.

2.3 Kalman Filter

See [9, 16, 22] for a more comprehensive treatment of Kalman filters. Kalman filters (KF) have traditionally been used in tracking. Unlike in least-squares regression, the KF maintains a state representation that is updated on each timestep. This makes the KF a type of state-space model. In a KF, the true location $\mathbf{x}_t$ of the object is assumed hidden and is modeled as a Gaussian random variable. Noisy observations $\mathbf{y}_t$ of the hidden state are assumed available. Transitions from one hidden state to another are modeled with a linear Gaussian function: the next location $\mathbf{x}_{t+1}$ is a linear function of the current location $\mathbf{x}_t$ plus some Gaussian noise $\mathbf{w}_t$ where $\mathbf{w} \sim N[0, Q]$. Observations are also assumed to be a linear function of the hidden state plus some Gaussian noise $\mathbf{v}_t$ where $\mathbf{v} \sim N[0, R]$ and $\mathbf{y}_1 \sim N[\mu, \Sigma]$. That is,

$$\begin{aligned}\mathbf{x}_{t+1} &= A\mathbf{x}_t + \mathbf{w}_t \\ \mathbf{y}_t &= B\mathbf{x}_t + \mathbf{v}_t\end{aligned}$$

Using the observations $\mathbf{y}_t$, the KF computes the probability of state $\mathbf{x}_t$ given the observations up to time t, that is $P(\mathbf{x}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_t)$. This is called filtering. The probability of a future state $\mathbf{x}_{t+k}$ given the current state $\mathbf{x_t}$, that is $P(\mathbf{x}_{t+k} \mid \mathbf{x}_t)$ can also be computed. This is called prediction. Finally, given future observations up to time $t+k$, the probability of the state at time t, $P(\mathbf{x}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t+k})$, can be computed. This is called smoothing. Filtering, prediction, and smoothing are all types of inference.

One of the major advantages of the KF is that the number of parameters needed to represent the state space does not explode as more observations are acquired. This is because the KF models the state as a Gaussian random variable and state transitions as a linear Gaussian function. Consequently, the state remains a Gaussian random variable, requiring only that the mean and the covariance be stored. This permits the KF to be compactly formulated using recursive updates. However, the dynamics of the object being tracked may not exactly satisfy the linear Gaussian assumption. For instance, the noise in the observations may not be Gaussian, or the object may be moving in a nonlinear fashion. The extended Kalman filter, the unscented Kalman filter, and the switching Kalman filter address the assumption of linear dynamics while particle filters additionally address the assumption of Gaussian noise. Figure 1 shows the DBN representation for the KF. Appendix A contains the update equations for the KF.

2.4 Switching Kalman Filter

See [2, 6, 10, 15] for a more comprehensive treatment of switching Kalman filters. Switching Kalman filters (SKF), also known as jump Markov linear systems, maintain a state representation that is updated on each timestep. Like with the KF, this also makes the SKF a type of state-space model. The basic difference between the SKF and KF is that the SKF can track objects with nonlinear dynamics while the KF can only track objects with linear dynamics. For instance, suppose that a KF is being used to track an object and the object is moving according to a dynamical model that can be approximated by the KF. Suppose that something then causes the dynamics of the object's movement to change, such as having to avoid an obstacle. A KF will initially predict poorly since the dynamics of the object have changed and a single KF can only encode one dynamical model. If it were possible, however, to recognize (or predict) when the object's dynamics were about to change, a different KF (encoding the new dynamics) could be used to track the object at that point. The SKF attempts to do exactly that. Essentially, an SKF computes a piece-wise linear approximation of the path of an object moving with nonlinear dynamics. In an SKF, the value of a switch variable indicates the particular KF being used on each time step. Like KFs, SKFs can also be formulated as a DBN. Figure 2 shows the DBN representation for the SKF. Appendix A contains the update equations for the SKF.

In an SKF, both the values of the state variable X and the values of the switch variable S are unknown. Thus, the KF that is generating observations at a given timestep is unknown. As a consequence, unlike in the KF, the problem of inferring the switch node or hidden state values in a SKF is intractable. For example, suppose we are observing an object whose dynamics can be described by switching among n KFs. At timestep 1, it is unknown which KF generated the observation, so the state distribution for each KF must be maintained, giving a belief state of size n. At timestep 2, which KF generated the observation is again unknown. Because the KF for timestep 1 is also unknown, the belief state is now of size n^2. Consequently, at timestep T, the belief state will be of size n^T. Thus, inference based on an exact state description is intractable for SKFs. Various approximate inference techniques have, however, been developed to address the problem of inference in SKFs and in general. We will focus on a method called the Generalized Pseudo Bayesian Algorithm [2, 10].

2.5 Generalized Pseudo Bayesian Algorithm

See [2, 10] for a more comprehensive treatment of the Generalized Pseudo Bayesian algorithm. The Generalized Pseudo Bayesian algorithm is an approximate inference method for the SKF. The essential idea of many of the approximate inference methods for SKFs is that if the values of the switch variables in the SKF were known, then the belief state would no longer be exponential in size and the Kalman filter equations could be used to efficiently compute the values of the state variables.

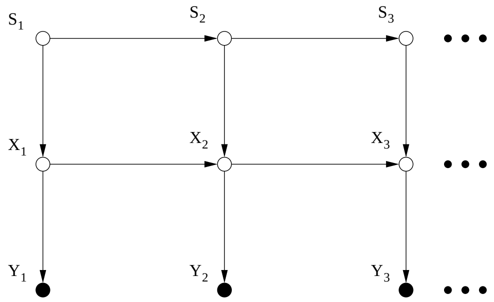


Figure 2: DBN formulation of the switching Kalman filter. S represents switches, X represents state, and Y represents observations. Clear nodes are hidden and shaded nodes are observed.

In comparison, rather than explicitly estimating switch node values, the Generalized Pseudo Bayesian algorithm works by weighting and summing over the hidden state values of each Kalman filter for those values that are more than k timesteps old. The weights are the probabilities that each Kalman filter generated the observation k timesteps ago. The number of timesteps k can be chosen, leading to an order k Generalized Pseudo Bayesian algorithm. Of course, the larger the k, the greater the cost of inference. Suppose $k = 2$ and let $\mathbf{x}^{ij}$ be the belief state if Kalman filter i is used on timestep $t-1$ and Kalman filter j is used on timestep t, let V^{ij} be the corresponding covariance, and let W^{ij} be the probability that the observation at timestep $t-1$ was generated using Kalman filter i and the observation at timestep t was generated using Kalman filter j. Then the order two generalized pseudo-Bayesian algorithm works by "collapsing" over the means and covariances two time steps ago as follows.

$$\begin{aligned}(\mathbf{x}^j, V^j) &= \text{Collapse}(\mathbf{x}^{ij}, V^{ij}, W^{ij}) \\ \mathbf{x}^j &= \sum_i W^{ij}\mathbf{x}^{ij} \\ V^j &= \sum_i W^{ij}V^{ij} + \sum_i W^{ij}(\mathbf{x}^{ij} - \mathbf{x}^j)(\mathbf{x}^{ij} - \mathbf{x}^j)'\end{aligned}$$

See Appendix A for more detailed equations of the Generalized Pseudo Bayesian algorithm with respect to the SKF. Other approximate inference methods for the SKF besides the Generalized Pseudo Bayesian algorithm include variational inference, sampling methods, and a Viterbi algorithm [16, 19].

3 Related Work on Tracking

We distinguish general statistical tracking techniques from storm tracking algorithms which may have a statistical foundation. Storm tracking methods can be divided into extrapolation and/or statistical algorithms, and other more knowledge-intensive algorithms. Extrapolation algorithms use deterministic linear models and can be loosely divided into centroid methods and correlation methods. Centroid methods work by identifying and tracking features associated with storm cells. The extrapolation algorithms have low complexity; their disadvantages are that they track the storm cell rather than the larger storm, resulting in less accurate long-term predictions, and that they have difficulty identifying when and how storms split.

The SCIT Algorithm [8] is one example of an extrapolation algorithm. It uses linear-least squares over the last five data points to predict storm locations; data points are the centroid locations of the storm cell being tracked. The TITAN methodology of [4] is another extrapolation algorithm, but it also uses cross-correlation. More knowledge-intensive algorithms for storm tracking include the Gandolf system developed by [20] which treats storm cells as objects and then models the meteorologic evolution of each object; and the Growth and Decay Storm Tracker [24] which focuses on tracking the encompassing storm using an elliptical filter, rather than tracking a storm cell.

Unlike the previous methods, rather than working with storm centroids, [11, 12] work with radar images. Given a sequence of weather images (e.g., radar reflectivity), [11, 12] first use K-means to identify storm clusters at different levels of granularity. The motion estimates for clusters in an image at time t are then computed by finding the corresponding regions of pixels for the image at time $t+1$ that minimize error. These estimates are then smoothed using a Kalman filter. Doing this for different levels of granularity has the advantage of permitting both longer-term and shorter-term forecasts.

In comparison, the Ensemble Kalman filter [5, 7, 14, 18] maintains a state representation of the atmosphere and evolves the state according to atmospheric dynamics. Predictions can then be made using the resulting model. A single state consists of $(latitude, longitude)$ and the accompanying features for that location such as temperature and surface pressure for different atmospheric levels. Because the state represents the atmosphere, the size of the state vector is large, and estimating the covariance using the standard Kalman filter is computationally expensive. To address this, ensemble-based techniques like the Ensemble Kalman filter were developed in which an ensemble of trajectories are used to estimate the covariance. Unlike the Kalman filter, the Ensemble Kalman filter can approximate nonlinear dynamics. Increasing the number of ensemble members in the Ensemble Kalman filter decreases the root mean square error in the forecast but increases the computational cost [7, 18].

Work on tracking within the machine learning community is primarily statistical: techniques such as Kalman filters, extended Kalman filters, switching Kalman filters, and particle filters in particular, are commonly used. Traditional tracking applications include tracking of ballistic objects such as missiles, and tracking of enemy aircraft [21]. Other applications include tracking vehicles [13] and tracking people [17, 19]. For instance, switching Kalman filters have been used to track humans alternating between running and walking [19]. While tracking in sensor networks is based on statistical techniques, e.g., using Kalman filters [17] or the extended Kalman filter [3], the distributed aspect of the sensor network introduces interesting variations on the tracking problem such as distributed versus centralized tracking [3].

4 Problem Formulation and Solution

This paper addresses the problem of tracking storm cells over time. The larger context for this research will use the predicted storm locations to identify future radar configurations. We constrain the problem in two ways. First, we are only interested in tracking a previously identified storm cell; we assume that there are meteorological algorithms that peform identification. Second, we assume that storm cell observations are obtained at regular intervals. This is not a necessary assumption, but it simplifies prediction using the Kalman filter.

Two types of approaches can be taken to address the problem of tracking storm cells over time: meteorological and statistical. A meteorology-based approach, such as the Ensemble Kalman filter [5], would model the atmospheric conditions, and predict the storm location based on that. Alternatively, a more purely statistical approach, such as least-squares regression [8] or Kalman filters, first solves the meteorological problem of storm centroid identification and then performs tracking, taking the storm location information as a given. We choose the statistical approach for several reasons. First, we are interested in satisfying real-time constraints, such as computing a new prediction every thirty seconds as in the CASA network. In this case, the Ensemble Kalman filter is too computationally expensive. Second, we are interested in investigating how well a purely statistical approach can perform.

To provide a comparison, we examine three statistical approaches. The first is the SCIT algorithm [8] which uses least-squares regression over the previous five storm locations. The second is the basic Kalman filter. The third is the switching Kalman filter which allows a piece-wise linear approximation to storm tracks. However, while least-squares linear regression [8] has been previously used to do storm cell tracking, to our knowledge, neither Kalman filters nor switching Kalman filters have been used to predict the locations of storm centroids. Linear least-squares regression provides a standard statistical approach already in use for storm tracking to which we can compare the Kalman filter and the switching Kalman filter.

5 Experiments

This section presents results comparing how well the KF, the SKF, and linear least-squares regression (SCIT [8]) perform in tracking a storm and predicting its future location.

5.1 Implementation

We implemented the equations in Appendix A for the SKF model in C++ and using the matrix library CwMtx and the GNU Scientific library. The SKF model was used to do prediction in both the basic KF (by setting the number of KFs to one) and the SKF. The regress function in Matlab was used for linear least-squares regression.

5.2 Data

Real storm track data was obtained from NSSL (National Severe Storms Laboratory), courtesy of Kurt Hondl and the WDSS-II simulator. The data consists of 35 storm tracks ranging in length from ten to 30 data points, identified using the SCIT algorithm [8]. Each track is a sequence of latitude and longitude coordinates with observations occurring approximately every five minutes. We note that SCIT, which stands for Storm Cell Identification and Tracking, both identifies and tracks storms. Each new datapoint that the SCIT algorithm acquires must be assigned to either an existing storm track or to a new storm track. A regression model is then fit to the five most recent datapoints in the track and used to predict future datapoints for that track. Our data consists of the storm tracks produced by SCIT.

5.3 Inference and Prediction

Inference in the SKF was done using a second order Generalized Pseudo Bayesian algorithm: i.e., collapsing was done over states two timesteps and older. This required computing the terms $\Pr(S_{t-1} = i, S_t = j | y_{1:t})$ and $\Pr(S_t = j | y_{1:t})$, where S_t represents the switch node value at time t and $y_{1:t}$ represents the sequence of observations from time 1 to t; see Appendix A for more details. These terms permit us to compute the most likely sequence of switch node values with $\text{argmax}_j P(S_t = j | S_{t-1} = i, e_{1:t})$ beginning at time 1 and ending at time t. In comparison to the Viterbi algorithm (see [16, 22] for a description), this algorithm is simpler because we have already computed the terms $\Pr(S_{t-1} = i, S_t = j | y_{1:t})$ and $\Pr(S_t = j | y_{1:t})$. The switch node value at t indicates the KF that will give the best prediction, hence that KF was used to do the 1-step prediction. Prediction with linear least-squares was done by using the previous five points to compute the line with the smallest least-squares error. The average x-interval was then added to the most recent x-coordinate to get the next x-coordinate. Least-squares was then used to determine the corresponding y-coordinate.

5.4 Training

We compared hand-initializing the parameters with using Expectation-Maximization (EM) [15] to learn the parameters for the KF and the SKF models. See Appendix A for a description of the model parameters. We defined each observation to be a vector of length two comprised of a *(latitude, longitude)* pair. We defined the mean of each hidden state to be a vector of length four comprised of the 4-tuple, *(latitude, longitude, latitude velocity, longitude velocity)*. We hand-coded the parameters as follows. The KF used generated points to the northeast. That is, the parameter matrix A was,

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The first column represents latitude, the second represents longitude, the third represents latitude velocity, and the fourth represents longitude velocity. The first row encodes the linear update equation for the latitude, the second row encodes the update equation for the longitude, the third row encodes the update equation for latitude velocity, and the fourth row encodes the update equation for longitude velocity. For instance, the latitude at time $t+1$ will be the latitude for the hidden state at time t, since there is a 1 in row 1 column 1, plus the latitude velocity for the hidden state at time t, since there is a 1 in row 1 column 3. Similarly, longitude at time $t+1$ will be the longitude for the hidden state at time t, since there is a 1 in row 2 column 2, plus the longitude velocity for the hidden state at time t, since there is a 1 in row 2 column 4. Consequently, if there is little noise and no observations, then iterating this KF will generate points to the northeast.

The SKF used four KFs; three generated points to the north, east, and northeast respectively and the fourth encoded a stationary system. For both the KF and the SKF, the first observation $\mathbf{y}_1$ was used for the initial state μ. For the SKF, the prior and transitions for the switches were uniform, i.e., all KFs had equal probability of being used. Below are the parameters for the SKF. The KF parameters are those for $s = 1$. The remaining hand-coded parameters for all $1 \leq s \leq 4$ are,

$$\Sigma_s = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \quad Q_s = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \quad R_s = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \quad B_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We now present the learned KF parameters obtained using EM. Due to the limited number of tracks, parameter learning was done with leave one out cross-validation: i.e., we trained on 34 tracks and tested on the 35^{th}. We did this 35 times so that we could test on each of the tracks. The learned KF parameters below are from a representative training run on 34 tracks.

$$\mu = \begin{bmatrix} 32.2006 \\ -94.58 \\ 0.06057 \\ -0.00713 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 10.5616 & -6.40907 & -0.01063 & 0.04784 \\ -6.40907 & 4.48945 & 0.01336 & -0.03375 \\ -0.01063 & 0.01336 & 9.03106e-05 & -0.00010 \\ 0.04784 & -0.03375 & -0.00010 & 0.00025 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.04520 & 0.01209 & -0.00075 & -0.00074 \\ 0.01209 & 0.02358 & -0.00279 & 0.00204 \\ -0.00075 & -0.00279 & 0.07144 & 0.00319 \\ -0.00074 & 0.00204 & 0.00318 & 0.06414 \end{bmatrix} \quad R = \begin{bmatrix} 0.00885 & 0.00124 \\ 0.00124 & 0.00663 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.99518 & -0.00177 & 0.30253 & 0.05275 \\ 0.00311 & 1.00081 & 0.05589 & 0.20061 \\ -0.00354 & -0.00121 & 0.47183 & -0.01887 \\ 0.001878 & 0.00049 & 0.00159 & 0.48156 \end{bmatrix} \quad B = \begin{bmatrix} 0.99867 - 0.00049 - 0.00676 - 0.00677 \\ -3.10335e - 061.00007 - 0.013060.01299 \end{bmatrix}$$

The μ and Σ parameters represent the mean and covariance of the initial state respectively; the first row/column represents latitude, the second represents longitude, the third represents latitude velocity, and the fourth represents longitude velocity. The Q parameters represent the covariance for hidden state while the R parameters represent the covariance for the observations.

5.5 Results

Table 1 summarizes the average root mean-square error (RMSE) for the latitude and longitude error over all storm tracks for each of the methods. Table 2 contains the individual latitude and longitude 1-step and 2-step prediction RMSEs for each storm track for each of the methods. The RMSE was calculated using only predicted points. For 1 step prediction, the first predicted point was the 6th point; for 2 step prediction, the first predicted point was the 7th point. Figures 3 to 9 plot the 1-step predictions for linear least-squares regression and the hand-coded SKF for fourteen different storm tracks. Table 3 summarizes the average computation time required for making a 1-step prediction. Figure 10 shows how the computation time required for making a 1-step prediction changes according to the number of previous observations. Computation times were recorded for a 1 GHz Dell Optiplex GX150.

	Average 1-Step RMSE		Average 2-Step RMSE	
Method	Latitude	Longitude	Latitude	Longitude
KF	0.2248	0.1923	0.3359	0.2871
KF-EM	0.1967	0.1680	0.2680	0.2389
SKF	0.1914	0.1702	0.2642	0.2421
SKF-EM	0.2577	0.2070	0.3948	0.3110
Least-squares	0.2114	0.2107	0.3030	0.3081

Table 1: Statistical summary of the error in the results. KF corresponds to using the hand-coded KF parameters. KF-EM corresponds to using the learned KF parameters. SKF corresponds to using the hand-coded SKF parameters. SKF-EM corresponds to using the learned SKF parameters.

5.6 Discussion

Although linear least-squares regression (SCIT [8]) was originally used to identify the storm tracks from raw data, Table 1 shows that the average track RMSE is lowest for the learned KF and hand-coded SKF models. Similar results hold for both the 1-step and 2-step prediction. Because of the limited amount of training data, we will focus on the results for the hand-coded SKF. In comparison to Table 1, Table 2 shows that the individual track RMSE is sometimes lower for the SKF and sometimes lower for the least-squares method. Additionally, the SKF may have lower latitude error but higher longitude error, or vice versa. We therefore compare the different methods based on the track RMSE instead of the average RMSE. For instance, Figures 3, 4, and 5 show 1-step predictions for tracks 4, 5, 7, 21, 28, and 29 for which the SKF has lower RMSE, while Figures 6, 7, and 8 show 1-step predictions for tracks 9, 13, 17, 19, 25, and 30 for which least-squares has lower RMSE. The most apparent difference is that the tracks in Figures 3, 4, and 5 are much less linear than most of the tracks in Figures 6, 7, and 8, which would explain why least-squares had higher RMSE than the SKF for tracks 4, 5, 7, 21, 28, and 29. We note that track 25 in Figure 8b is an exception to this: the reason that the least-squares method predicts better on this track may be that the nonlinearity is primarily due to a single outlying point.

Examining the prediction errors in Table 2 shows that for a given method, the latitude RMSE is often higher than the longitude RMSE, particularly for the Kalman filter methods. This may be a function of the storm dynamics or measurement technique, since the discrepancy is most apparent for tracks with large RMSE such as tracks 0 and 1 (Figure 9). We note that the least-squares method seems to have less of a bias between the latitude and longitude RMSEs. This is particularly apparent if the average RMSEs for least-squares in Table 1 are examined. It should be noted that 0.1 degrees of latitude corresponds to 6.9 miles amd that 0.1 degrees of longitude corresponds to at most 6.9 miles [1].

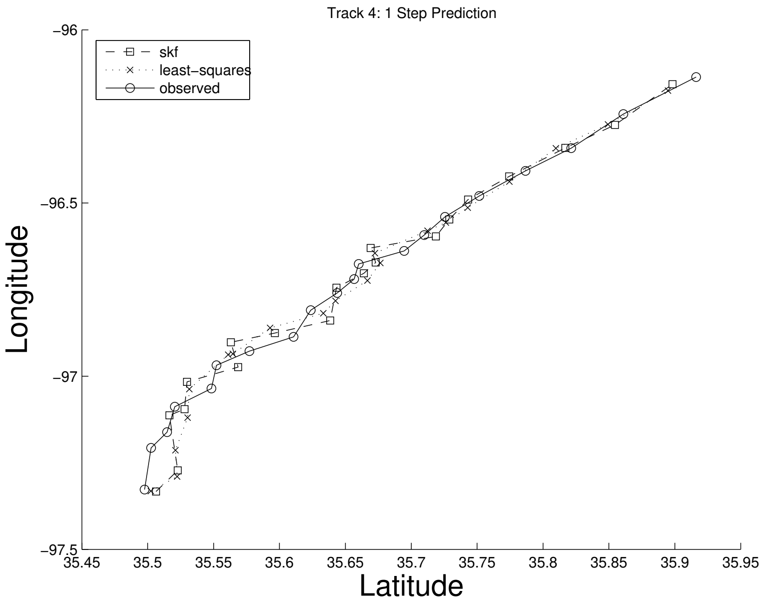
The distribution of the prediction errors in Table 2 for the SKF and least-squares varies widely, from about 0.88 to about 0.008. Examining the actual tracks shows that tracks with large errors, such as tracks 0 and 1 in Figure 9, often looked like they were two different tracks joined together. The tracks were originally identified using SCIT so this phenomenon may be a result of how new datapoints were associated with existing tracks.

Track (length)	KF latitude , longitude	KF-EM	SKF	SKF-EM	Least-Squares
1-step RMSE					
0(29)	0.90403 , 0.87084	0.66329 , 0.62988	0.65358 , 0.62568	0.79970 , 0.62534	0.8844 0.8105
1(26)	0.28037 , 0.20774	0.23183 , 0.17729	0.24253 , 0.18008	0.25189 , 0.18060	0.2625 0.1943
2(19)	0.93394 , 0.44150	0.79020 , 0.37823	0.81639 , 0.38751	0.78763 , 0.37498	0.8863 0.4001
3(13)	0.95617 , 0.03500	0.94850 , 0.02406	0.95633 , 0.03498	0.88765 , 0.13087	0.9575 0.0335
4(25)	0.01270 , 0.02374	0.03605 , 0.03289	0.01291 , 0.02361	0.17607 , 0.09288	0.0129 0.0298
5(15)	0.03839 , 0.02909	0.06536 , 0.02718	0.03907 , 0.02814	0.29582 , 0.14171	0.0362 0.0514
6(13)	0.17919 , 0.18751	0.02561 , 0.05743	0.05044 , 0.06549	0.14970 , 0.04959	0.1574 0.2630
7(10)	0.02895 , 0.02852	0.06693 , 0.02856	0.02876 , 0.02846	0.28579 , 0.13384	0.0308 0.0524
8(10)	0.78352 , 0.68303	0.56678 , 0.57628	0.58890 , 0.68814	0.50042 , 0.53172	0.5381 0.5516
9(21)	0.01519 , 0.04505	0.06511 , 0.03324	0.01575 , 0.04630	0.28808 , 0.12940	0.0139 0.0325
10(10)	0.59466 , 0.51433	0.46582 , 0.45963	0.51105 , 0.44573	0.47369 , 0.47804	0.5886 0.4599
11(14)	0.15919 , 0.06070	0.15464 , 0.04646	0.14955 , 0.06149	0.16126 , 0.06325	0.1457 0.0550
12(10)	0.49354 , 0.10549	0.46926 , 0.05324	0.48260 , 0.11151	0.46865 , 0.06393	0.4879 0.0916
13(10)	0.02071 , 0.02703	0.07400 , 0.01906	0.02106 , 0.02809	0.22175 , 0.08205	0.0191 0.0201
14(16)	0.03286 , 0.10896	0.03265 , 0.10510	0.03327 , 0.10879	0.07695 , 0.11480	0.0352 0.1140
15(18)	0.18437 , 0.30524	0.15048 , 0.26642	0.16155 , 0.27157	0.16179 , 0.28451	0.1614 0.2859
16(15)	0.13117 , 0.01848	0.13943 , 0.02827	0.13074 , 0.01861	0.14179 , 0.07134	0.1304 0.0190
17(30)	0.01374 , 0.02069	0.04573 , 0.02150	0.01415 , 0.02133	0.23476 , 0.10871	0.0132 0.0200
18(13)	0.40717 , 0.47234	0.35426 , 0.42316	0.36260 , 0.41330	0.40053 , 0.46698	0.3876 0.4645
19(10)	0.02153 , 0.02646	0.09471 , 0.02003	0.02301 , 0.02696	0.27751 , 0.10670	0.0202 0.0203
20(13)	0.16809 , 0.63656	0.14896 , 0.54184	0.16866 , 0.55059	0.14391 , 0.53838	0.1577 0.9725
21(15)	0.02338 , 0.01362	0.01922 , 0.01425	0.01230 , 0.01156	0.11763 , 0.05779	0.0532 0.1206
22(14)	0.02617 , 0.02951	0.05842 , 0.03927	0.02675 , 0.02824	0.29816 , 0.16510	0.0227 0.0623
23(16)	0.45009 , 0.24595	0.38711 , 0.21272	0.38679 , 0.20938	0.44089 , 0.20279	0.4226 0.2576
24(10)	0.01786 , 0.00990	0.03208 , 0.01974	0.01733 , 0.00998	0.06088 , 0.06560	0.0190 0.0080
25(10)	0.02985 , 0.02077	0.03465 , 0.03870	0.02981 , 0.02093	0.06812 , 0.09432	0.0302 0.0137
26(14)	0.01580 , 0.01088	0.02427 , 0.01245	0.01574 , 0.01095	0.03863 , 0.05903	0.0141 0.0146
27(10)	0.95479 , 0.57028	0.69257 , 0.62705	0.72265 , 0.57324	0.61171 , 0.67966	0.7024 0.7652
28(13)	0.00791 , 0.00945	0.02710 , 0.03502	0.00789 , 0.00923	0.04195 , 0.07488	0.0088 0.0174
29(13)	0.01230 , 0.00876	0.01482 , 0.01351	0.01177 , 0.00921	0.06883 , 0.05681	0.0135 0.0674
30(13)	0.01492 , 0.01681	0.01440 , 0.02211	0.01514 , 0.01650	0.07028 , 0.07780	0.0134 0.0157
31(10)	0.06778 , 0.58171	0.06100 , 0.59681	0.06831 , 0.58205	0.08084 , 0.60920	0.0666 0.5813
32(12)	0.01935 , 0.02829	0.02973 , 0.02441	0.01971 , 0.02785	0.04196 , 0.04637	0.0162 0.0388
33(12)	0.02278 , 0.00968	0.01348 , 0.00868	0.02514 , 0.01051	0.08191 , 0.06407	0.0251 0.0111
34(12)	0.06853 , 0.52020	0.08299 , 0.43387	0.06863 , 0.44096	0.07129 , 0.42925	0.0631 0.4579
2-step RMSE					
0(29)	1.21630 , 1.16825	0.68660 , 0.65311	0.67704 , 0.64370	1.13030 , 0.70503	1.2531 1.1228
1(26)	0.46442 , 0.35365	0.32500 , 0.25578	0.35797 , 0.27886	0.36081 , 0.25621	0.4116 0.3038
2(19)	1.55785 , 0.72952	1.14146 , 0.54824	1.22024 , 0.57454	1.06590 , 0.51886	1.4223 0.6290
3(13)	1.03041 , 0.05767	1.00827 , 0.02731	1.03104 , 0.05606	0.93559 , 0.23321	1.0288 0.0407
4(25)	0.02140 , 0.04147	0.06429 , 0.05698	0.02217 , 0.03991	0.34313 , 0.16398	0.0217 0.0435
5(15)	0.06325 , 0.06003	0.12730 , 0.04740	0.06570 , 0.05697	0.53662 , 0.23308	0.0582 0.0599
6(13)	0.53560 , 0.49325	0.31926 , 0.29397	0.39247 , 0.29854	0.44746 , 0.21463	0.3420 0.5171
7(10)	0.06899 , 0.05102	0.13009 , 0.05176	0.06818 , 0.05422	0.51347 , 0.22948	0.0553 0.0703
8(10)	0.51934 , 0.49061	0.49765 , 0.48225	0.52141 , 0.49047	0.47213 , 0.50752	0.5223 0.4914
9(21)	0.02487 , 0.04928	0.12667 , 0.04047	0.02677 , 0.05084	0.52822 , 0.21615	0.0232 0.0327
10(10)	1.37708 , 0.73216	0.71964 , 0.71881	0.86063 , 0.74285	0.63517 , 0.72373	1.2383 0.7596
11(14)	0.22529 , 0.06231	0.24085 , 0.05558	0.22436 , 0.05999	0.27363 , 0.11780	0.2232 0.0615
12(10)	0.66553 , 0.28999	0.57397 , 0.16130	0.60490 , 0.29649	0.50800 , 0.07360	0.6822 0.2054
13(10)	0.03417 , 0.02922	0.15374 , 0.02300	0.03510 , 0.03020	0.41116 , 0.15374	0.0193 0.0112
14(16)	0.07222 , 0.11667	0.06825 , 0.11164	0.07503 , 0.11659	0.17330 , 0.15400	0.0559 0.1259
15(18)	0.30239 , 0.47156	0.21437 , 0.38572	0.24052 , 0.38035	0.21945 , 0.40947	0.2367 0.4083
16(15)	0.14292 , 0.02086	0.15521 , 0.03946	0.14294 , 0.02023	0.19603 , 0.12593	0.1424 0.0219
17(30)	0.02245 , 0.03220	0.08806 , 0.03803	0.02380 , 0.03303	0.44403 , 0.19025	0.0193 0.0275
18(13)	0.72440 , 0.82926	0.54281 , 0.65352	0.57500 , 0.64629	0.61420 , 0.72000	0.6445 0.7806
19(10)	0.06300 , 0.04511	0.20541 , 0.02676	0.06435 , 0.03875	0.49823 , 0.18391	0.0205 0.0212
20(13)	0.27815 , 1.09454	0.21750 , 0.80421	0.27851 , 0.84291	0.22710 , 0.78427	0.2469 1.6092
21(15)	0.11383 , 0.05213	0.03335 , 0.02971	0.06979 , 0.03586	0.25556 , 0.12695	0.1166 0.2114
22(14)	0.04048 , 0.05615	0.11356 , 0.06803	0.04336 , 0.05422	0.54115 , 0.26928	0.0303 0.0836
23(16)	0.76208 , 0.41350	0.58048 , 0.30809	0.58811 , 0.30848	0.66237 , 0.28729	0.6831 0.4180
24(10)	0.03532 , 0.02249	0.05931 , 0.04017	0.03345 , 0.01703	0.17247 , 0.16005	0.0404 0.0197
25(10)	0.05657 , 0.03452	0.05897 , 0.07224	0.05604 , 0.03328	0.17893 , 0.19264	0.0568 0.0193
26(14)	0.03479 , 0.03125	0.04081 , 0.01619	0.03565 , 0.02640	0.09685 , 0.10385	0.0151 0.0228
27(10)	1.30375 , 1.06167	0.79899 , 1.10806	0.82747 , 1.04791	0.72851 , 1.18679	0.6853 1.1375
28(13)	0.01823 , 0.02175	0.05025 , 0.06594	0.01719 , 0.02344	0.12785 , 0.14669	0.0185 0.0290
29(13)	0.03021 , 0.02611	0.03549 , 0.01906	0.02896 , 0.02337	0.15989 , 0.10311	0.0302 0.0961
30(13)	0.02288 , 0.02322	0.01800 , 0.03887	0.02446 , 0.02092	0.13731 , 0.14404	0.0158 0.0182
31(10)	0.07782 , 0.64456	0.06418 , 0.67883	0.08182 , 0.64521	0.17148 , 0.70915	0.0735 0.6512
32(12)	0.03468 , 0.06071	0.05122 , 0.04099	0.03651 , 0.05707	0.12645 , 0.10659	0.0269 0.0694
33(12)	0.03896 , 0.02060	0.01897 , 0.01181	0.04451 , 0.02151	0.15502 , 0.12039	0.0428 0.0146
34(12)	0.11406 , 0.64969	0.11687 , 0.62723	0.11506 , 0.64953	0.16352 , 0.62551	0.1033 0.6494

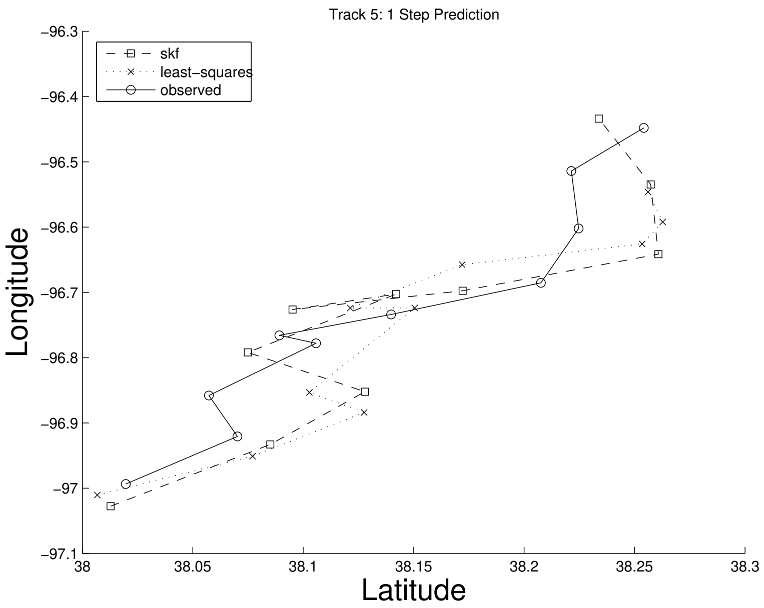
Table 2: Root mean square error of individual tracks for 1-step and 2-step predictions.

Number of KFs	Computation Time for 1-Step Prediction in Seconds		
	Average	Maximum	Minimum
1	0.000155	0.000864	0.000103
4	0.001689	0.006479	0.001138
8	0.006655	0.028375	0.004553

Table 3: Computation time required for making a 1-step prediction according to the number of Kalman filters used.

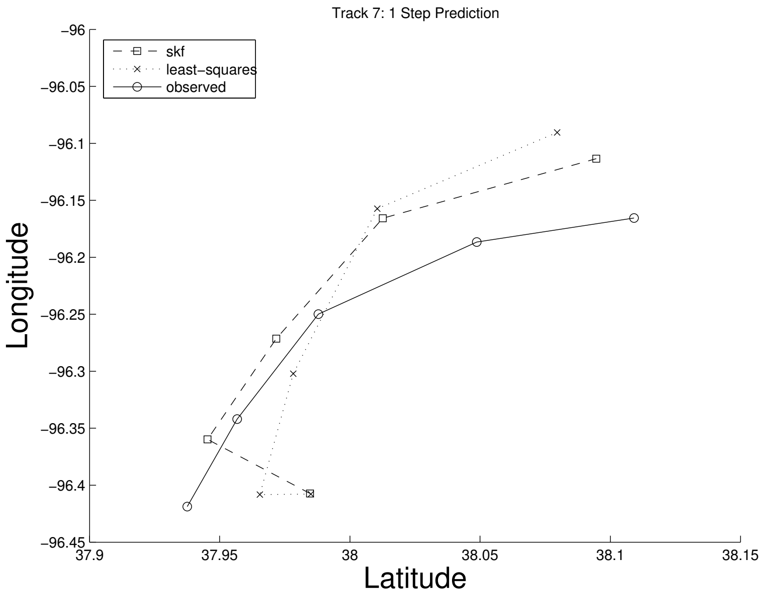


(a) Track 4

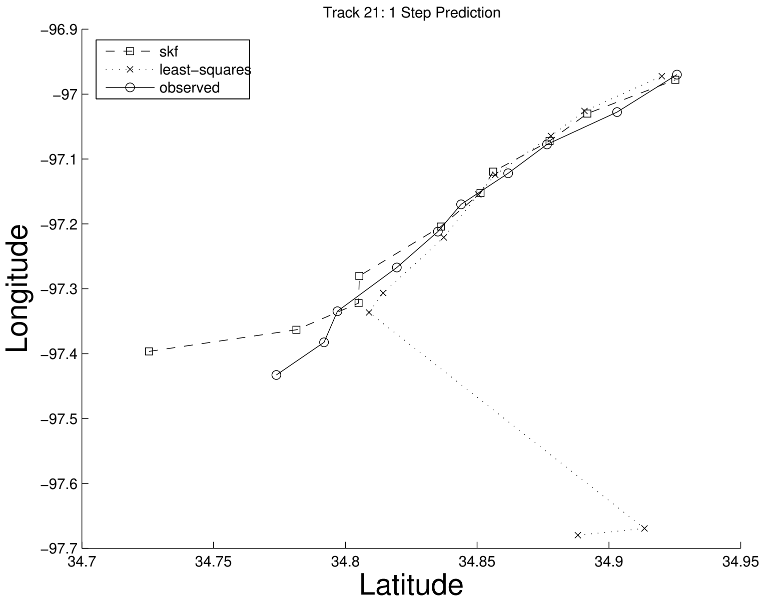


(b) Track 5

Figure 3: 1-Step predictions for storm tracks 4 and 5. Switching Kalman filter parameters were hand-coded.

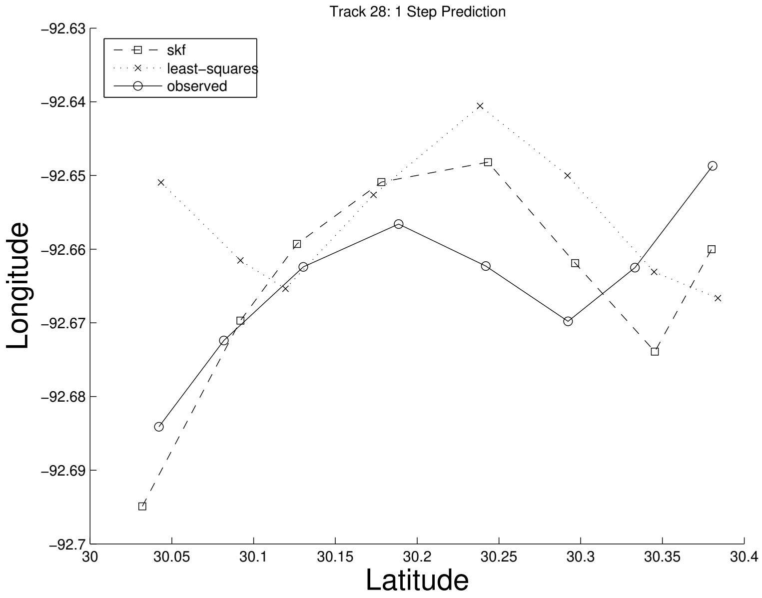


(a) Track 7

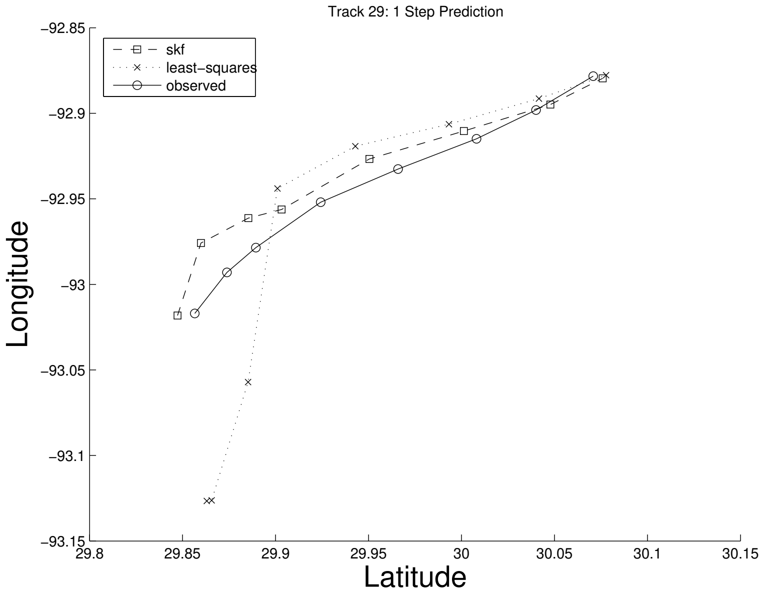


(b) Track 21

Figure 4: 1-Step predictions for storm tracks 7 and 21. Switching Kalman filter parameters were hand-coded.

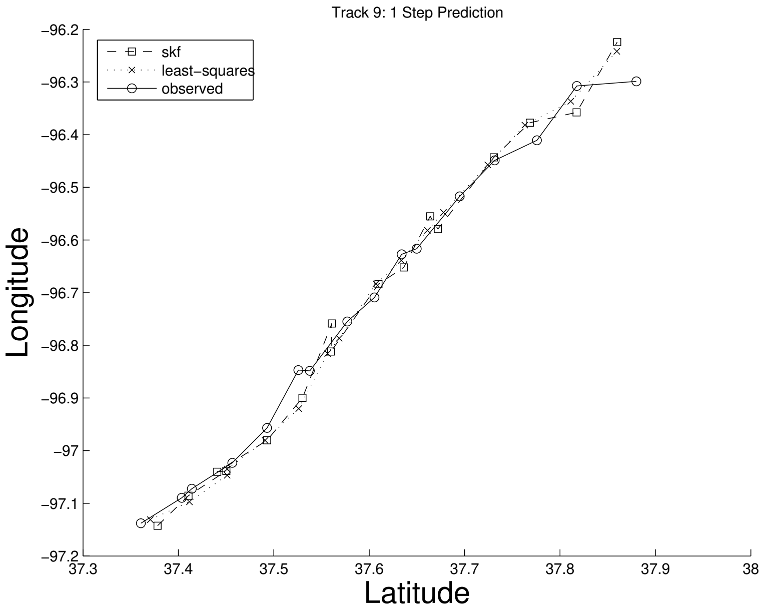


(a) Track 28

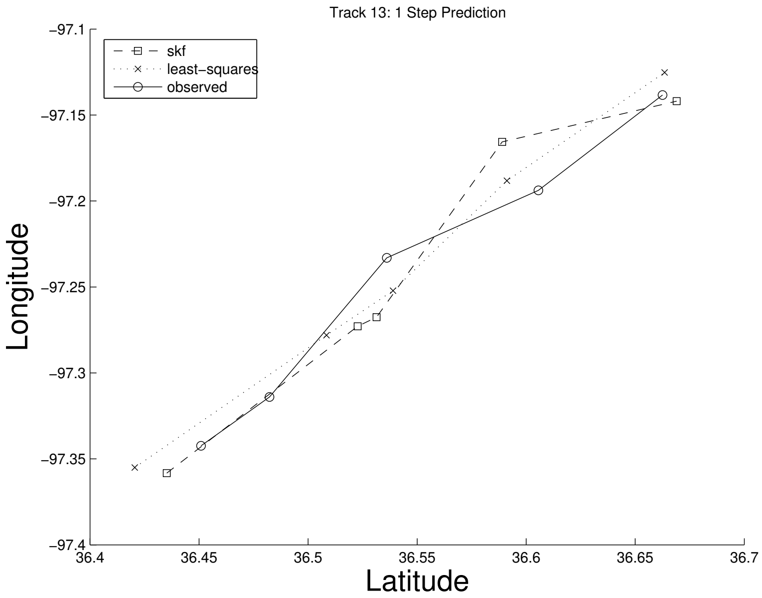


(b) Track 29

Figure 5: 1-Step predictions for storm tracks 28 and 29. Switching Kalman filter parameters were hand-coded.

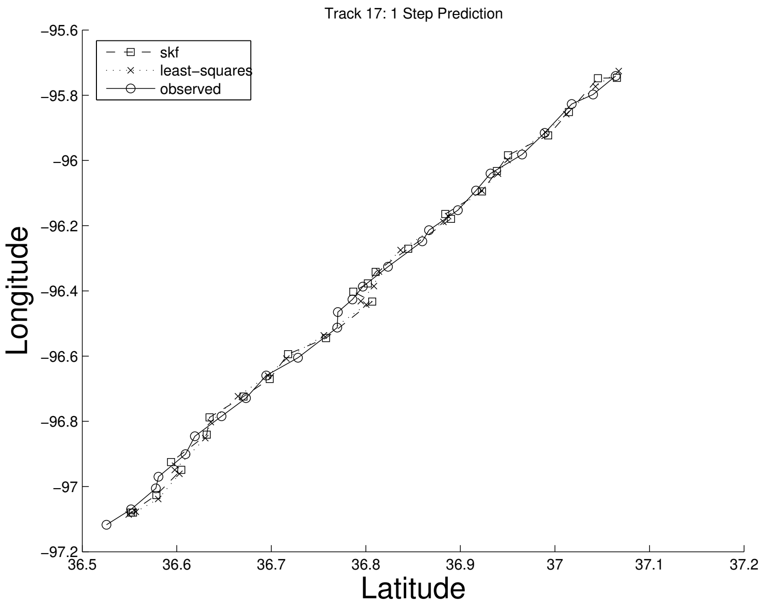


(a) Track 9

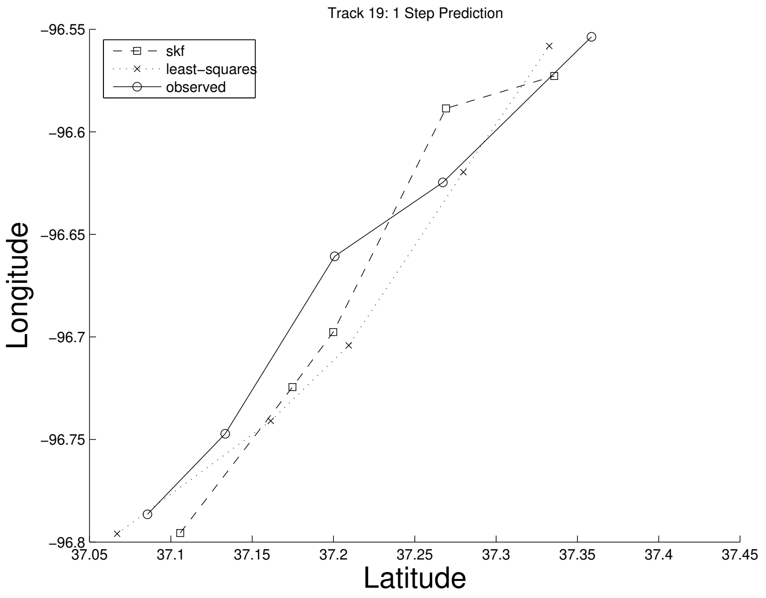


(b) Track 13

Figure 6: 1-Step predictions for storm tracks 9 and 13. Switching Kalman filter parameters were hand-coded.

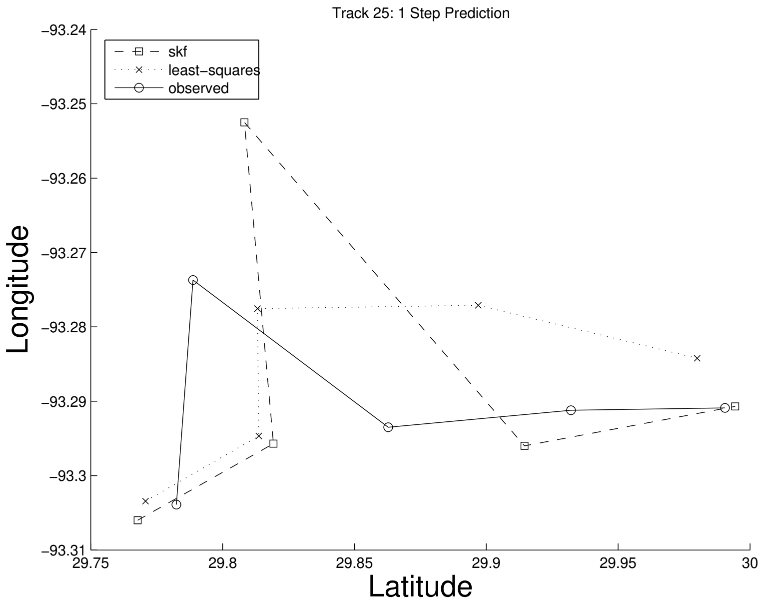


(a) Track 17

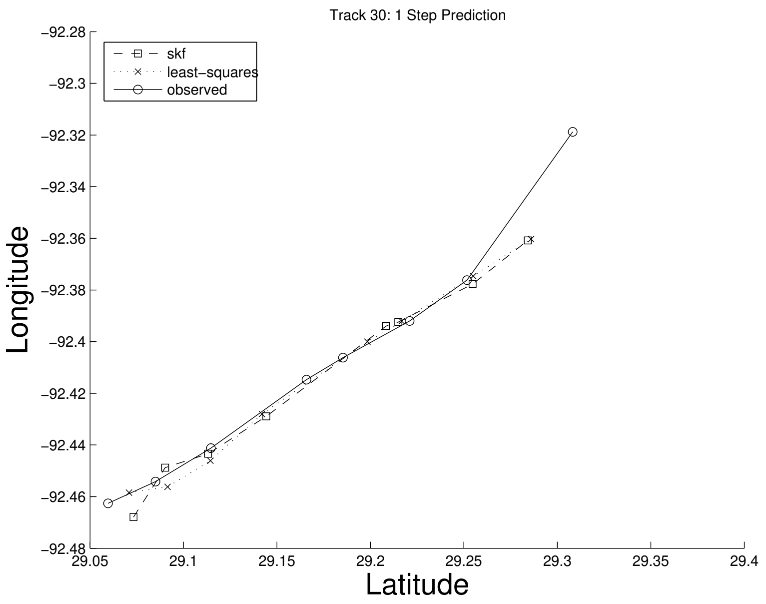


(b) Track 19

Figure 7: 1-Step predictions for storm tracks 17 and 19. Switching Kalman filter parameters were hand-coded.

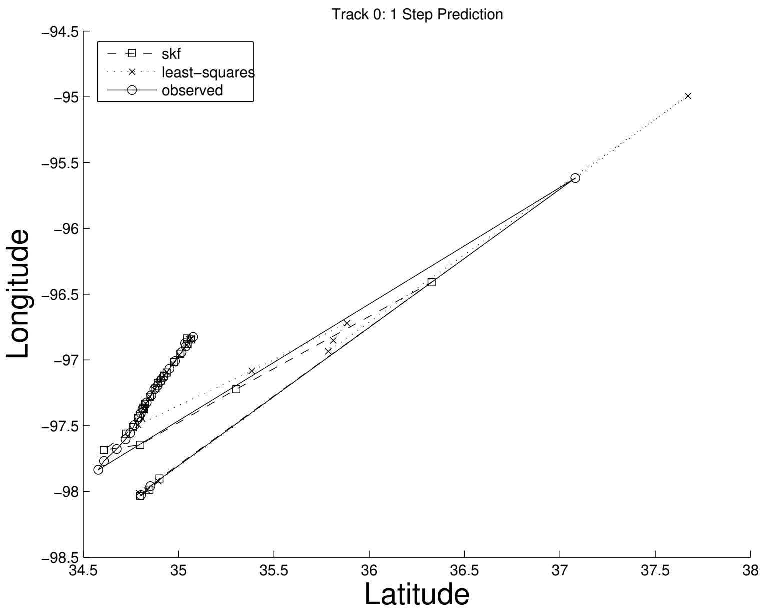


(a) Track 25

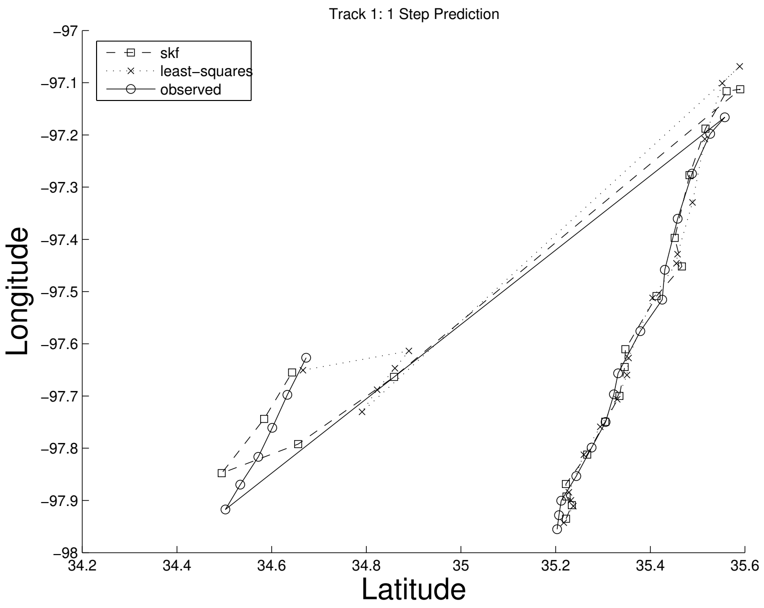


(b) Track 30

Figure 8: 1-Step predictions for storm tracks 25 and 30. Switching Kalman filter parameters were hand-coded.



(a) Track 0



(b) Track 1

Figure 9: 1-Step predictions for storm tracks 0 and 1. Switching Kalman filter parameters were hand-coded.

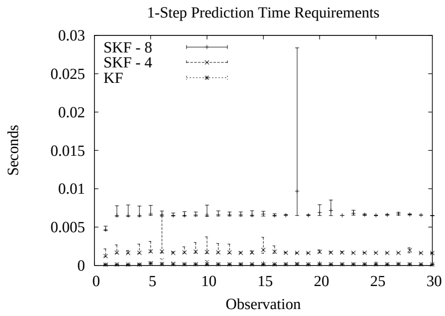


Figure 10: Computation time required for making a 1-step prediction.

The SKF framework could be used instead of SCIT to associate new datapoints with existing tracks or to identify new tracks.

The KF using learned parameters does well, about as well as the SKF using hand-coded parameters. This is one indication that the poor performance of the KF using hand-coded parameters is due in part to improperly initialized parameters. One caveat is that there were a limited number of tracks for training, hence the use of leave-one-out cross-validation. More tracks would help verify the performance of the KF and SKF with learned parameters. We note that the SKF using learned parameters performs worse than the KF with hand-coded parameters. This may be due in part to overfitting as a result of the small dataset. Work by [6] discusses the difficulties with training SKFs, referred to in the paper as switching state-space models, in particular the difficulty in learning when to switch. What we found empirically was that the performance of the learned models depended strongly on how the model parameters were initialized before learning. For instance, although not shown, we found that with a slightly different parameter initialization (using $0.01 \cdot I$ instead of $0.1 \cdot I$ for the Q matrices where I is the identity matrix), and using 8 KFs in the SKF, gave improved performance for the learned SKF over the hand-coded SKF. Also note that the Kalman filter models need to condition on an adequate number of observations occurring sufficiently frequently to do well. With only a few observations or with infrequent observations, linear least-squares regression may have the advantage. We note that linear least-squares regression over only the last five data points is a sort of piece-wise linear approximation. But unlike in the SKF model, linear least-squares regression can only switch to an arbitrarily sloped new line segment every five points.

Finally, Table 3 shows that while increasing the number of Kalman filters used in the SKF increases the computation time required for making a 1-step prediction, the computation time required is still sub-second. Figure 10 shows that the computation time required for making a 1-step prediction is not affected by the number of previous observations. With respect to other storm-tracking methods besides least-squares, we note that the hiearchical clustering method in [11] uses data acquired every 5-6 minutes and requires thirty seconds of computation to perform prediction. In comparison, the ensemble Kalman filter makes long-term (12 hour) forecasts [14].

6 Summary

For certain types of tracks, the SKF appears to have the potential to better predict the future location of a storm centroid than the linear least-squares SCIT [8] algorithm. Since we compared the SKF and linear least-squares regression on data already processed by SCIT, a comparison of the two methods on raw data would be useful. Some of the difficulties initially anticipated with tracking storms were not encountered. For instance, storms are not discrete objects, but tracking the storm centroid seems to have avoided addressing this issue. However, accounting for some of the variability of a storm by tracking the entire storm cell and any encompassing larger or neighboring storms using additional hidden variables in the graphical model could still potentially improve predictions.

7 Future Work

There are several directions for future research. With respect to the problem of tracking meteorological phenomena, one interesting direction would be to look at multiple target-tracking: e.g., tracking the splitting and merging of storms. This would eliminate the reliance on SCIT for associating new datapoints with existing storm tracks or for identifying new storm tracks. Another interesting direction would be to incorporate more meteorological information into the SKF model, possibly by adding more hidden nodes. For instance, knowledge of the average velocity of a storm or the calibration error of the radars could be incorporated into the parameters. With respect to the models used, in this paper we focused on two related DBNs, the KF and the SKF. We are interested in exploring related models, such as an extended version of the SKF with multiple higher level layers instead of only a single switch layer. Such models would potentially permit multiscale predictions similar to those of [11, 12]. Since exact inference in models like the SKF is often computationally expensive, exploring approximate inference methods for such models would also be useful. Finally, it would be interesting to use the predictions made by models like the KF and SKF for decision-making.

References

[1] Latitude and longitude. *http://core.ecu.edu/geology/woods/LatLong.htm.*

[2] Y. Bar-Shalom. *Estimation and tracking : principles, techniques, and software.* Artech House, Boston, Massachusetts, 1993.

[3] R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91:8:1163–1171, 2003.

[4] M. Dixon and G. Weiner. TITAN: Thunderstorm identification, tracking analysis and nowcasting a radar based methodology. *J. Atmos. Ocean. Tech.*, 10:785–797, 1993.

[5] G. Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[6] Z. Ghahramani and G. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:831–864, 2000.

[7] P. Houtekamer and H. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126:796–811, 1998.

[8] J. Johnson, P. MacKeen, A. Witt, E. DeWayne Mitchell, G. Stumpf, M. Eilts, and K. Thomas. The storm cell identification and tracking algorithm: An enhanced WSR-88D algorithm. *Weather and Forecasting*, 13:263–276, 1998.

[9] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation.* Prentice Hall, Upper Saddler River, New Jersey, 2000.

[10] C-J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60:1–22, 1994.

[11] V. Lakshmanan. *A hierarchical, multiscale texture segmentation algorithm for real-world scenes.* PhD thesis, University of Oklahoma, 2001.

[12] V. Lakshmanan, R. Rabin, and V. DeBrunner. Multiscale storm identification and forecast. *Journal of Atmospheric Research*, pages 367–380, 2003.

[13] V. Lesser, C. Ortiz, and M. Tambe editors. *Distributed Sensor Networks: A multiagent perspective.* Kluwer Academic Publishers, Netherlands, 2003.

[14] H. Mitchell and P. Houtekamer. An adaptive ensemble Kalman filter. *Monthly Weather Review*, 126:416–433, 1998.

[15] K. Murphy. Switching Kalman filters. *Technical report, U. C. Berkeley*, 1998.

[16] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, Computer Science Division, 2002.

[17] N. Nguyen, H. Bui, S. Venkatesh, and G. West. Multiple camera coordination in a surveillance system. *ACTA Automatica Sinica*, 29:408–422, 2003.

[18] E. Ott, B. Hunt, I. Szunyogh, A. Zimin, E. Kostelich, M. Corazza, E. Kalnay, D. Patil, and J. Yorke. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus*, 2004.

[19] V. Pavlovic, J. Rehg, T-J. Cham, and K. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Intl. Conf on Computer Vision*, 1999.

[20] C. E. Pierce, P. J. Hardaker, C. G. Collier, and C. M. Haggett. GANDOLF: a system for generating automated nowcasts of convective precipitation. *Meteorol. Appl.*, 7:341–360, 2000.

[21] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, Boston, Massachusetts, 2004.

[22] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, 2nd Edition.* Prentice Hall, Upper Saddler River, New Jersey, 2003.

[23] G. Strang. *Introduction to Linear Algebra, 3rd Edition.* Wellesley-Cambridge Press, Wellesley, Massahcusetts, 2003.

[24] M. Wolfson, B. Forman, R. Hallowell, and M. Moore. The growth and decay storm tracker. In *American Meteorological Society 79th Annual Conference*, 1999.

[25] B. Xu, C.K. Wikle, and N.I. Fox. A kernel-based spatio-temporal dynamical model for nowacasting radar precipitation. *Submitted to the J. Amer. Stat. Assoc.*, 2003.

A Switching Kalman Filter Equations

The following are approximate inference and learning equations for the switching Kalman filter, taken directly from [15]. The equations are for a single sample of length T. Multiple samples are accommodated by looping. Approximate inference is done using the order two Generalized Pseudo Bayesian algorithm. By setting the number of Kalman filters to be one, we obtain the basic Kalman filter. Informally, $\mathbf{y}$ refers to the observation mean vector, $\mathbf{x}$ refers to the hidden state mean vector, V refers to the hidden state covariance matrix, e refers to the error, K refers to the Kalman gain, S refers to the switch nodes, M refers to the switch node probabilities, and L refers to the likelihood of an observation. In the following equations, it is assumed that the initial state μ and initial state covariance Σ for each Kalman filter are known, and that the initial switch distribution π and transition switch distribution Z, are known. The π parameters represent the prior distribution over the KFs. Each row i represents a KF. The value stored in row i is the probability that the ith KF generates the first observation. The Z parameters represent transition probabilities for the switch nodes of the SKF. Each row and column represents a KF. The value stored in row i and column j is the probability that the ith KF generated the observation at time $t-1$ and the jth KF generated the observation at time t. The matrices A, B, Q, and R for each Kalman filter are also assumed known, see Section 2.3 for definitions. We define n to be the number of Kalman filters in the model. The E-Step of Expectation-Maximization corresponds to the filtering and smoothing equations, the M-Step corresponds to the maximization equations. The equations from [15] are then,

Notation:

$$\begin{aligned}
\mathbf{x}_{t|\tau}^{i(j)} &= E[\mathbf{X}_t|\mathbf{y}_{1:\tau}, S_{t-1}=i, S_t=j] \\
\mathbf{x}_{t|\tau}^{j(k)} &= E[\mathbf{X}_t|\mathbf{y}_{1:\tau}, S_t=j, S_{t+1}=k] \\
\mathbf{x}_{t|\tau}^{j} &= E[\mathbf{X}_t|\mathbf{y}_{1:\tau}, S_t=j] \\
V_{t|\tau}^{j} &= \mathrm{Cov}[\mathbf{X}_t|\mathbf{y}_{1:\tau}, S_t=j] \\
V_{t,t-1|\tau}^{j} &= \mathrm{Cov}[\mathbf{X}_t, \mathbf{X}_{t-1}|\mathbf{y}_{1:\tau}, S_t=j] \\
V_{t,t-1|\tau}^{i(j)} &= \mathrm{Cov}[\mathbf{X}_t, \mathbf{X}_{t-1}|\mathbf{y}_{1:\tau}, S_{t-1}=i, S_t=j] \\
M_{t-1,t|\tau}(i,j) &= \Pr(S_{t-1}=i, S_t=j|y_{1:\tau}) \\
M_{t,\tau}(j) &= \Pr(S_t=j|y_{1:\tau}) \\
L_t^j &= (\mathbf{y}_t|\mathbf{y}_{1:t-1}, S_t=j)
\end{aligned}$$

Initialization, for $j = 1:n$:

$$\mathbf{x}_{1|0}^{j} = \mu(j)$$

$$
\begin{aligned}
V^j_{1|0} &= \Sigma(j) \\
M_{0|0}(j) &= \pi(j)
\end{aligned}
$$

Filtering, for $t = 1 : T$, for $j, i = 1 : n$:

$$
\begin{aligned}
(\mathbf{x}^{i(j)}_{t|t}, V^{i(j)}_{t|t}, V^{i(j)}_{t,t-1|t}, L^{i(j)}_t) &= \text{Filter}(\mathbf{x}^i_{t-1|t-1}, V^i_{t-1|t-1}, \mathbf{y}_t; A_j, B_j, Q_j, R_j) \\
M_{t-1,t|t}(i,j) = \Pr(S_{t-1} = i, S_t = j | y_{1:t}) &= \frac{L_t(i,j) Z(i,j) M_{t-1|t-1}(i)}{\sum_i \sum_j L_t(i,j) z(i,j) M_{t-1|t-1}(i)} \\
M_{t|t}(j) &= \sum_i M_{t-1,t|t}(i,j) \\
W^{i|j} &= \Pr(S_{t-1} = i | S_t = j, \mathbf{y}_{1:t}) = M_{t-1,t|t}(i,j)/M_{t|t}(j) \\
(\mathbf{x}^j_{t|t}, V^j_{t|t}) &= \text{Collapse}(\mathbf{x}^{i(j)}_{t|t}, V^{i(j)}_{t|t}, W^{i|j}_t)
\end{aligned}
$$

Smoothing, for $t = T-1 : 1$, for $j, k = 1 : n$:

$$
\begin{aligned}
(\mathbf{x}^{(j)k}_{t|T}, V^{(j)k}_{t|T}, V^{j(k)}_{t+1,t|T}) &= \text{Smooth}(\mathbf{x}^k_{t+1|T}, V^k_{t+1|T}, \mathbf{x}^j_{t|t}, V^j_{t|t}, V^k_{t+1|t+1}, V^{j(k)}_{t+1,t|t+1}; A_k, Q_k) \\
U^{j|k}_t &= \Pr(S_t = j | S_{t+1} = k, \mathbf{y}_{1:T}) \approx \frac{M_{t|t}(j) Z(j,k)}{\sum_{j'} M_{t|t}(j') Z(j',k)} \\
M_{t,t+1|T}(j,k) &= U^{j|k}_t M_{t+1|T}(k) \\
M_{t|T}(j) &= \sum_k M_{t,t+1|T}(j,k) \\
W^{k|j}_t &= \Pr(S_{t+1} = k | S_t = j, \mathbf{y}_{1:T}) = M_{t,t+1|T}(j,k)/M_{t|T}(j) \\
(\mathbf{x}^j_{t|T}, V^j_{t|T}) &= \text{Collapse}(x^{(j)k}_{t|T}, V^{(j)k}_{t|T}, W^{k|j}_t) \\
(\mathbf{x}_{t|T}, V_{t|T}) &= \text{Collapse}(x^j_{t|T}, V^j_{t|T}, W^j_t) \\
\mathbf{x}^{j(k)}_{t+1|T} &= \text{E}[\mathbf{x}_{t+1} | \mathbf{y}_{1:T}, S_{t+1} = k, S_t = j] \approx \mathbf{x}^k_{t+1|T} \\
V^k_{t+1,t|T} &= \text{CollapseCross}(\mathbf{x}^{j(k)}_{t+1|T}, \mathbf{x}^{(j)k}_{t|T}, V^{j(k)}_{t+1,t|T}, U^{j|k}_t) \\
\mathbf{x}^{()k}_{t|T} &= \text{E}[\mathbf{X}_t | \mathbf{y}_{1:T}, S_{t+1} = k] = \sum_j \mathbf{x}^{(j)k}_{t|T} U^{j|k}_t \\
V_{t+1,t|T} &= \text{CollapseCross}(\mathbf{x}^k_{t+1|T}, \mathbf{x}^{()k}_{t|T}, V^k_{t+1,t|T}, M_{t+1|T}(k))
\end{aligned}
$$

Filter function:

$$
\begin{aligned}
(\mathbf{x}_{t|t}, V_{t|t}, V_{t,t-1|t}, L_t) &= \text{Filter}(\mathbf{x}_{t-1|t-1}, V_{t-1|t-1}, \mathbf{y}_t; A, B, Q, R) \\
\mathbf{x}_{t|t-1} &= A \mathbf{x}_{t-1|t-1} \\
V_{t|t-1} &= A V_{t-1|t-1} A' + Q \\
\mathbf{e}_t &= \mathbf{y}_t - B \mathbf{x}_{t|t-1} \\
S_t &= B V_{t|t-1} B' + R \\
K_t &= V_{t|t-1} B' S_t^{-1} \\
L_t &= N(\mathbf{e}_t; 0, S_t) \\
\mathbf{x}_{t|t} &= \mathbf{x}_{t|t-1} + K_t \mathbf{e}_t \\
V_{t|t} &= (I - K_t B) V_{t|t-1} = V_{t|t-1} - K_t S_t K_t' \\
V_{t,t-1|t} &= (I - K_t B) A V_{t-1|t-1}
\end{aligned}
$$

Smooth function:

$$
\begin{aligned}
(\mathbf{x}_{t|T}, V_{t|T}, V_{t+1,t|T}) &= \text{Smooth}(\mathbf{x}_{t+1|T}, V_{t+1|T}, \mathbf{x}_{t|t}, V_{t|t}, V_{t+1|t+1}, V_{t+1,t|t+1}; A, Q) \\
\mathbf{x}_{t+1|t} &= A \mathbf{x}_{t|t}
\end{aligned}
$$

$$
\begin{aligned}
V_{t+1|t} &= AV_{t|t}A' + Q \\
\mathbf{e}_t &= \mathbf{y}_t - B\mathbf{x}_{t|t-1} \\
J_t &= V_{t|t}A'V_{t+1|t}^{-1} \\
\mathbf{x}_{t|T} &= \mathbf{x}_{t|t} + J_t(\mathbf{x}_{t+1|T} - \mathbf{x}_{t+1|t}) \\
V_{t|T} &= V_{t|t} - J_t(V_{t+1|T} - V_{t+1|t})J_t' \\
V_{t+1,t|T} &= V_{t+1,t|t+1} + (V_{t+1|T} - V_{t+1|T} - V_{t+1|t+1})V_{t+1|t+1}^{-1}V_{t+1,t|t+1}
\end{aligned}
$$

Collapse function:

$$
\begin{aligned}
(\mathbf{x}, V) &= \text{Collapse}(\mathbf{x}^j, V^j, W^j) \\
\mathbf{x} &= \sum_j W^j \mathbf{x}^j \\
V &= \sum_j W^j V^j + \sum_j W^j (\mathbf{x}^j - \mathbf{x})(\mathbf{x}^j - \mathbf{x})'
\end{aligned}
$$

CollapseCross function:

$$
\begin{aligned}
(\mathbf{x}, \mathbf{y}, V) &= \text{CollapseCross}(\mathbf{x}^j, \mathbf{y}^j, V^j, W^j) \\
\mathbf{x} &= \sum_j W^j \mathbf{x}^j \\
\mathbf{y} &= \sum_j W^j \mathbf{y}^j \\
V &= \sum_j W^j V^j + \sum_j W^j (\mathbf{x}^j - \mathbf{x})(\mathbf{y}^j - \mathbf{y})'
\end{aligned}
$$

Maximization equations for $i = 1 : n$ and m training samples:

Define:

$$
\begin{aligned}
W_t^j &= \Pr(S_t = j|\mathbf{y}_{1:T}) \\
\hat{\mathbf{x}}_t &= \hat{E}[\mathbf{x}_t] = \mathbf{x}_{t|T} \\
P_t &= \hat{E}[\mathbf{x}_t\mathbf{x}_t'] = V_{t|T} + \mathbf{x}_{t|T}\mathbf{x}_{t|T}' \\
P_{t,t-1} &= \hat{E}[\mathbf{x}_t\mathbf{x}_{t-1}'] = V_{t,t-1|T} + \mathbf{x}_{t|T}\mathbf{x}_{t-1|T}'
\end{aligned}
$$

Then,

$$
\begin{aligned}
\mu_i &= \frac{\sum_m \hat{x}_1}{\sum_m W_1^i} \\
\Sigma_i &= \frac{\sum_m W_1^i \hat{\mathbf{x}}_1 \hat{\mathbf{x}}_1' - \mu_i (\sum_m W_1^i \hat{\mathbf{x}}_1') - (\sum_m W_1^i \hat{\mathbf{x}}_1)\mu_i') + (\sum_m W_1^i)\mu_i\mu_i'}{\sum_m W_1^i} \\
A_i &= \left(\sum_m \sum_{t=2}^{T} W_t^i P_{t,t-1}\right)\left(\sum_m \sum_{t=2}^{T} W_t^i P_{t-1}\right)^{-1} \\
B_i &= \left(\sum_m \sum_{t=1}^{T} W_t^i \mathbf{y}_t \hat{\mathbf{x}}_t'\right)\left(\sum_m \sum_{t=1}^{T} W_t^i P_t\right)^{-1} \\
Q_i &= \left(\frac{1}{\sum_m \sum_{t=2}^{T} W_t^i}\right)\left(\sum_m \sum_{t=2}^{T} W_t^i P_t - A_i \sum_m \sum_{t=2}^{T} W_t^i P_{t,t-1}'\right) \\
R_i &= \left(\frac{1}{\sum_m \sum_{t=1}^{T} W_t^i}\right)\sum_m \sum_{t=1}^{T} W_t^i (\mathbf{y}_t\mathbf{y}_t' - C_i \hat{\mathbf{x}}_t \mathbf{y}_t')
\end{aligned}
$$

$$
\begin{aligned}
Z(i,j) &= \frac{\sum_m \sum_{t=2}^{T} \Pr(S_{t-1}=i, S_t=j|\mathbf{y}_{1:T})}{\sum_m \sum_{t=1}^{T-1} W_t^i} \\
\pi &= \frac{1}{m}\sum_m W_1^i
\end{aligned}
$$