Efficient Tracking and Querying for Coordinated Uncertain Mobile Objects

Nicholas D. Larusso, Ambuj Singh
Department of Computer Science, University of California, Santa Barbara
{nlarusso, ambuj}@cs.ucsb.edu

Abstract— **Accurately estimating the current positions of moving objects is a challenging task due to the various forms of data uncertainty (e.g. limited sensor precision, periodic updates from continuously moving objects). However, in many cases, groups of objects tend to exhibit similarities in their movement behavior. For example, vehicles in a convoy or animals in a herd both exhibit tightly coupled movement behavior within the group. While such statistical dependencies often increase the computational complexity necessary for capturing this additional structure, they also provide useful information which can be utilized to provide more accurate location estimates.**

In this paper, we propose a novel model for accurately tracking coordinated groups of mobile uncertain objects. We introduce an exact and more efficient approximate inference algorithm for updating the current location of each object upon the arrival of new (uncertain) location observations. Additionally, we derive probability bounds over the groups in order to process probabilistic threshold range queries more efficiently. Our experimental evaluation shows that our proposed model can provide 4X improvements in tracking accuracy over competing models which do not consider group behavior. We also show that our bounds enable us to prune up to 50% of the database, resulting in more efficient processing over a linear scan.

I. Introduction

The availability of cheap location sensors has made it possible to track the position of anything that moves. For example, services such as Loopt [1] and Google Latitude [2] allow users to view their current location in relation to the (real-time) locations of their friends. Additionally, movement patterns are critical to zoologists for understanding animal flocking behaviors [1] as well as migratory patterns [2]. To manage the large volumes of location and mobility data, new techniques are required which efficiently track, index and query the locations of mobile objects.

Unfortunately, much of the available location and movement data remains highly uncertain. In addition to low sensor quality, several causes for data uncertainty include power limitations, bandwidth constraints, and intermittent signal disruptions [3], [1], [4]. These limitations make it essential for the spatiotemporal database system to appropriately manage data uncertainty. This includes appropriately answering queries over *probability distributions* as opposed to point locations [5], [6], as well as intelligently utilizing all known information to update and refine the location probability for each moving object. For example, when a mobile phone is unable to make a connection with a satellite to obtain a GPS reading, the phone's location may be estimated by triangulating the signal using the cellular towers, which provides a spatial region over which the phone is likely to be located. Utilizing this uncertain information is critical for applications that wish to provide robust, uninterrupted service.

The combination of *temporal* and *location* uncertainty makes it difficult to provide meaningful query results over the locations of mobile objects. Thus, if data uncertainty is not properly managed, it can compound, rendering query results useless. To address this issue, a tracking model can be integrated into the database which effectively integrates uncertain location information with the dynamics of the mobile object to provide an estimate of current and future locations as well as model the inherent uncertainty. Previous work has shown that this tracking can be improved further by exploiting any additional known structure from the problem (e.g. road network) [7], [8]. However, it is not always possible to obtained such a structured set of constraints which dictate object movement (e.g. tracking animals in the wild). In these cases, it may be possible to extract a different kind of structure which may help us provide accurate localization for these objects.

In this paper, we investigate the case of managing groups of mobile objects with coordinated movements. For example, vehicles traveling in a convoy or flocks of animals tend to move as a group. We propose a novel tracking model which exploits the group structure in the mobile objects in order to provide more accurate location estimates (and predictions). Our model is based on the principles of the Kalman filter model (KFM), which allows us to scale to large groups of mobile objects efficiently. We further utilize the group structure to provide probability bounds on the location of the mobile objects. From this bound, we develop a pruning technique which allows us to efficiently process probabilistic threshold range queries.

More specifically, we assume that we have a set of M moving objects, each of which sends location updates to a centralized server at regular intervals, although updates may be missing at random. Updates only contain information about the object's location, and may be uncertain. The objective in this work is to *accurately resolve the true location of the set of mobile objects and provide a meaningful measure of uncertainty in our estimate*. Using a more accurate description

[1]https://www.loopt.com/
[2]https://www.google.com/latitude/
[3]https://www.openstreetmap.org/

of the objects' locations and their uncertainty translates to improved query results which we use to effectively answer probabilistic threshold range queries (PTRQs).

We model the location uncertainty with a Gaussian distribution with a fixed variance, R, which describes the sensor error. This allows us to efficiently store our complete knowledge of an object's location with a mean vector and a covariance matrix. The mean vector is the size of the belief state, in our case this is a vector of four values: $[loc_x, loc_y, vel_x, vel_y]$ and the covariance matrix contains all of the uncertainty information. In total, the space required to maintain the current uncertain location for all objects in the database is $O(4N + \frac{4(4+1)}{2}N)$ (to maintain the complete trajectory simply multiply this by the number of time steps, T). The result of our tracking model is a probabilistic trajectory which accurately describes the trajectory of each object as well as where we are uncertain of the object's location. This output may be used to answer queries in a probabilistic spatiotemporal database.

In this paper, we make the following contributions:

- We introduce a novel model for tracking coordinated groups of uncertain mobile objects (section III). Our model, *ClustKFM*, utilizes the group structure to improve the location estimates of each individual mobile object while scaling to large groups.
- We derive an exact inference algorithm for *ClustKFM* which allows us to update our location estimate for each object by considering all available information. We also introduce a novel deterministic approximate inference algorithm which scales linearly with the size of the group of dependent objects.
- We develop a pruning technique for efficient probabilistic threshold range queries (section IV). Using the group structure from our tracking model, we derive a bounding region which is guaranteed to encompass a specified amount of probability mass from each object.
- Lastly, we provide a thorough experimental evaluation of our proposed tracking model and apply it to two real datasets (section V). We examine the tracking accuracy, resulting querying quality, as well as the computational gains we achieve from our pruning technique.

II. Preliminaries

The Kalman filter has a long history in the field of tracking mobile objects [9], however, previous studies and applications of the KFM do not consider utilizing group behavior to reduce location uncertainty as we do in this work. Before developing our model, in this section, we briefly introduce the basic concepts of the original KFM for tracking moving objects. For further details about the Kalman filter, we refer the reader to [10], [11]. Table I lists our notation.

The Kalman Filter model (KFM) is a special case of the Hidden Markov Model (HMM) [9], [11], a dynamic probabilistic state space model. In the KFM, the hidden state is described with a Gaussian distribution, observations are assumed to be normally distributed from the true state, and state transitions are described with a Gaussian random walk.

TABLE I
Model Notation

Notation	Explanation
M	Number of mobile objects with correlated movement (group size)
x_i^t	Belief state (hidden) for object i at time t
$x_{1..M}^t$	The set of belief states for all objects at time t
$z_i^{0..t}$	Observations for object i from time $0..t$
A	Deterministic transition matrix
H	Deterministic observation mask
Q	Covariance matrix for the movement dynamics. Describes uncertainty in transition model.
R	Observation covariance matrix. Describes sensor uncertainty.
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with expected value μ and covariance matrix Σ.

More formally, the conditional independence assumptions are described in Eq. 1 and 2.

$$p(\mathbf{x}^t|\mathbf{x}^{t-1}) = \mathcal{N}(\mathbf{A}\mathbf{x}^{t-1}, \mathbf{Q}) \quad (1)$$

$$p(\mathbf{z}^t|\mathbf{x}^t) = \mathcal{N}(\mathbf{H}\mathbf{x}^t, \mathbf{R}) \quad (2)$$

The transition probability of our belief state x^t depends on the deterministic transition matrix A and transition covariance matrix Q. To explain the ideas behind the Kalman filter, we consider a simple example in which we are tracking a moving object in one-dimension. The representation of our belief state, x, contains the object's current location and velocity. Therefore, the transition matrix $\mathbf{A}$, is defined to encode $x_{loc}^t = x_{loc}^{t-1} + x_{vel}^{t-1} * t$ and $x_{vel}^t = x_{vel}^{t-1}$, that is $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. The covariance matrix Q describes the noise in the dynamic process (e.g. head/tail winds or loss of power due to friction). Suppose that we are only able to observe the object's location at discrete time steps, but have no way of acquiring the velocity directly. In this case the deterministic observation mask $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$, and we must infer velocity completely from the location observations. The covariance matrix R, describes how much uncertainty exists in these observations.

These model restrictions enable the posterior distribution to be evaluated analytically, providing a tractable procedure for exact inference. Efficient inference is crucial in applications involving streaming data since new estimates must be computed quickly as new data arrives. The Kalman filter can be used for three inference tasks: (1) *prediction*, (2) *filtering* and (3) *smoothing*. *Prediction* infers the next belief state, x^{t+1} given the observations up to the current time, $z^{0..t}$. Prediction causes an increase in variance since there are no observations to support the predicted belief state. *Filtering* infers the current belief state, x^t, given observations up to the current time, $z^{0..t}$. Filtering inference is computed by first predicting the current state, and then correcting this estimate using the current observation. *Smoothing* infers a past belief state, x^τ, where $0 \leq \tau \leq t$, given observations up to the current time,

$z^{0..\tau..t}$. The basic inference steps for the KFM are illustrated in figure 1. The KFM provides an efficient and accurate approach

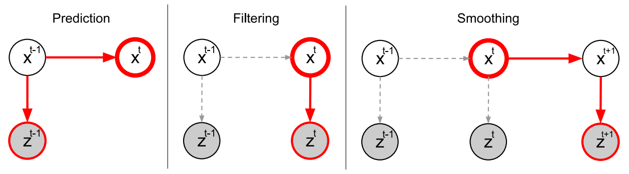


Fig. 1. KFM Inference steps: **Prediction** predicts the value of x^t from the last known position of the object, z^{t-1} and the movement model. This is computed in the graphical model by integrating out the unobserved value of x^{t-1}. **Filtering** *corrects* the value of x^t by incorporating the latest uncertain observation, z^t. *Smoothing* updates the filtered estimate for x^t by also incorporating information from later observations ($z^{1..T}$).

for tracking individual objects, however, it is unable to take advantage of dependencies between objects, making it more susceptible to the uncertainty pattern of the location updates. Next we introduce *ClustKFM* in detail, provide the algorithms necessary for exact and approximate inference, and show how to learn the model parameters.

III. Tracking Coordinated Uncertain Mobile Objects

In this section, we propose a model for tracking a coordinated group of moving objects with location uncertainty. Our model, the clustered Kalman filter (*ClustKFM*), is a generative probabilistic model, defined as a linear dynamical system in which the joint distribution over all variables is a multivariate Gaussian. This structure allows us to derive a set of equations to perform exact inference. While this procedure is relatively fast for medium sized groups (e.g. $10-50$), it does not scale well to large groups of uncertain mobile objects because of the required matrix inversion operation. To address this issue, we also introduce a novel approximate inference algorithm that scales linearly with the group size.

A. ClustKFM Model Definition

Figure 2 shows the graphical structure of *ClustKFM*. The shaded nodes are observed variables and the unshaded nodes are latent variables. In this context, the observed variables, z_i^t's, are the uncertain location updates that each object sends to the MOD. These 'observations' correspond to the mean of a Gaussian distribution, with known variance, defined over a spatial region that describes the degree of uncertainty in the position update. The unobserved variables, x_i^t's and y^t, are the object belief states (i.e. current estimate of location and velocity) and the shared group-state, respectively.

Before explaining the random variables and their dependency structure in more detail, we first outline the generative process of the model in algorithm 1. The intuition is that objects can share location and velocity information through the group parameter, y^t. In essence, y^t acts like a cluster centroid, where each object contributes to the value of this parameter, which in turn, is used to provide additional information about the locations of other objects. This 'sharing' constrains objects to move in a similar pattern and, when combined with the

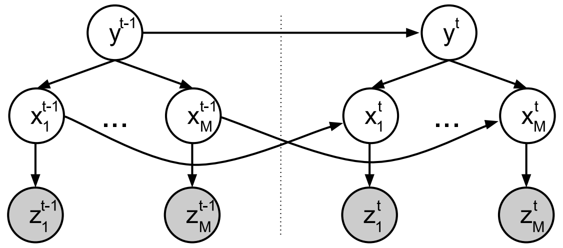


Fig. 2. Graphical structure of *ClustKFM*, our proposed model capable of exploiting the statistical dependencies between objects with clustered movement patterns.

Algorithm 1 Generative Process

```
// for each time step ...
for $t = 2 \to T$ do
    // draw new value of cluster state
    $y^t \sim N(y^{t-1}, W)$
    for $i = 1 \to M$ do
        // draw the position and velocity of the $i^{th}$ object
        $x_i \sim N(A(\alpha y^t + (1-\alpha)x_i^{t-1}), Q)$
        // draw a noisy observation for the $i^{th}$ object
        $z_i \sim N(Hx_i, R)$
    end for
end for
```

uncertain location updates, smooths our estimates to produce less jitter in the prediction of object movements.

$$p(z^t|x^t) = \mathcal{N}(Hx^t, R) \quad (3)$$
$$p(y^t|y^{t-1}) = \mathcal{N}(y^{t-1}, W) \quad (4)$$
$$p(x^t|y^t, x^{t-1}) = \mathcal{N}(A[\alpha y^t + (1-\alpha)x^{t-1}], Q) \quad (5)$$

Next, we introduce the functional form of the distributions used for the different aspects of the model: *observations*, *group dynamics* and *individual dynamics*. We start with the *observation* model in Eq. 3, which defines the probability of observing a specific location update given the current position of an object. This describes our knowledge that the location update pushed from each object to the server represents an uncertain location. The matrix, H, in this equation is simply a mask to note that we only observe the location of an object, not its velocity.

The *group dynamics* component is defined in Eq. 4. We define y to transition according to a Gaussian random walk with a diagonal covariance matrix $W = I_2\sigma_y^2$. This provides continuity of the group dynamics between consecutive time slices. Larger values of σ_y^2 allow the cluster of objects to change direction more quickly, while smaller values will restrict the group to maintain a very smooth trajectory. Gaussian linear dynamics provide a rather natural fit for our tracking application since moving objects are constrained by the laws of physics which restrict the amount with which they may change speed and direction over short periods of time. Lastly, the *individual dynamics* for each object is defined in Eq. 5. This equation defines how the location and velocity for an individual object change over time.

We define the state to be a mixture of the state at the previous time step, x_i^{t-1}, and the shared group parameter, y^t.

This can be viewed as a special case of Bayesian updating in which we use y^t as our prior and x_i^{t-1} as our observation. The parameter α, $0 \leq \alpha \leq 1$, determines how each of these terms are weighted. When $\alpha = 0$, the objects move completely independent of one another, while when $\alpha = 1$, the group of objects will all maintain exactly the same velocity. After we update the object state with the group information, we apply our (deterministic) transition matrix $\mathbf{A}$ to update the object's location. Given our conditional independence assumptions and distribution definitions, we show how to compute using our model.

B. Exact Inference

The three inferential problems we would like to solve in *ClustKFM* are that of filtering, prediction, and smoothing. Each of these inference procedures may be used in MODs and spatiotemporal databases, depending on the query time interval. For instance, in MODs, all queries are over the current and future positions of moving objects and therefore the tracking model will require both filtering and prediction. If historic queries are enabled in the database as well, then smoothing could be utilized to provide more accurate estimates of the object trajectories. Due to space restrictions, we do not discuss smoothing in this paper.

1) Filtering: The objective of filtering is to compute $p(x^t_{1..M}, y^t | z^t_{1..M})$ at time step t. Typically, inference in Bayesian networks is intractable due to the complicated integrals that are required to compute the distribution of interest and sampling approaches are usually required. However, we have taken careful consideration to ensure that exact inference is tractable by utilizing conjugate priors and defining our model to be jointly normally distributed. It is known that marginalizing a subset of random variables from a joint Gaussian distribution results in simply removing those elements from the expected value vector and the covariance matrix, resulting in a Gaussian distribution. Additionally, conditioning on any subset of variables of a joint Gaussian distribution also results in a Gaussian distribution [12]. Using these properties, and our conditional independence assumptions, we see that the distribution over a single time slice can be written as follows.

$$\begin{aligned} p(x^t_{1..M}, y^t, z^t_{1..M} | z^{t-1}_{1..M}) &= \int_{y^{t-1}, x^{t-1}_{1..M}} p(y^{t-1}, x^{t-1}_{1..M} | z^{t-1}_{1..M}) \\ &\times \prod_i^M p(z^t_i | x^t_i) p(x^t_i | y^t, x^{t-1}_i) \\ &\times p(y^t | y^{t-1}) dy^{t-1} dx^{t-1}_{1..M} \\ &= \mathcal{N}(^{-}_{x,y,z}, \mathbf{\Sigma}_{x,y,z}) \qquad (6) \end{aligned}$$

After receiving the location updates sent by each object, we can condition on the observed values of the z_i's and compute the posterior, $p(x^t_{1..M}, y^t | z^t_{1..M})$. Since this conditional is again Gaussian, we simply need to update our mean and covariance parameters to sufficiently represent the updated position of each object given the newly observed (uncertain) locations. The update equations for each moving object are given in Eq. 8 and 9.

$$\begin{aligned} K &= \Sigma_{x,y \rightarrow z} (\Sigma_{z,z})^{-1} && (7) \\ \mu^t_{filter} &= \mu^t_{x_{1..M}} + K(z^t_{1..M} - H\mu^t_{x_{1..M}}) && (8) \\ \Sigma^t_{filter} &= \Sigma_{x,y} - K(\Sigma_{x,y \rightarrow z})^T && (9) \end{aligned}$$

The idea in Eq. 8 is that we correct our prediction of the expected value of $x_{1..M}, y$ by accounting for the observed values of $z_{1..M}$. The correction is weighted by the covariance between the observations and the hidden variables times the precision (inverse covariance) of the observations. This allows us to consider the degree of uncertainty of the observations. If the uncertainty is high, then the weights computed in the gain matrix K will be very small since we have little confidence in these locations. Similarly, the covariance of the hidden variables is updated using the same two covariance terms. Note that we subtract K from our predicted covariance matrix, since covariance matrices are positive-semidefinite, this is guaranteed to result in a reduced covariance for the posterior. This is intuitive as we are updating our beliefs based upon new information (the observations of several location updates).

The problem of prediction is then to estimate the next state of each object given a sequence of observations up to the current time step. In our model, similarly to the Kalman filter, we implement a linear model of object movement. Therefore, the predicted expected location of an object is computed according to Eq. 5 where the value used for y^t is also predicted. Since we do not observe any additional information to confirm our estimate for the next time step, the variance increases as we predict further into the future. How quickly the variance grows depends on A, α, Q, T, as well as the current variance for x^{t-1} and y^t. Specifically, $\Sigma^t_{pred} = A(\alpha P^t_y \alpha^T + (1-\alpha) P^{t-1}_x (1-\alpha)^T) A^T + Q$.

C. Approximate Inference Algorithm

Because our model was defined to be jointly normally distributed, we can use the fact that the form of the posterior distribution over the hidden variables is also normal. Since the marginalization of a variable from a multivariate normal results in a normal distribution, we know that the form of the marginalized posterior distribution for y and for each x_i will also take the form of a normal. We use this observation to develop an efficient procedure for approximate inference.

The key idea behind our approximate inference algorithm is to take advantage of the conditional independence assumption between the x_i given the value of y. If we assume that the value of y has been observed, then the x_i's become conditionally independent and we can compute each distribution, $p(x^t_i | y^t)$ independently. Our approach works by splitting the inference into two steps in which we first approximate the posterior of y^t and then use this updated value to approximate the x_i's independently. An overview of our algorithm is represented pictorially in figure 3. We have applied this approximation to both the filtering and smoothing inference problems, though, due to space constraints we only discuss filtering in this paper. We approximate the updated value of y using Eq. 10. To clarify our assumption that y is observed and

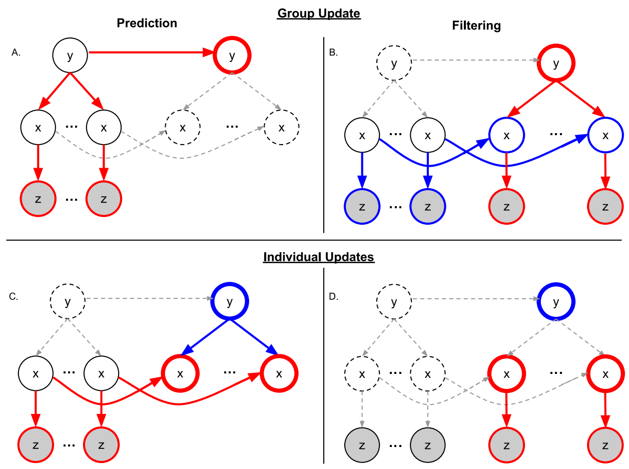


Fig. 3. *ClustKFM* approximate filtering inference steps: **A.** We predict the value for y^t from the previous observations and the group dynamics. **B.** The filtering for y^t requires that we first update (predict) the x_i^t's (blue arrows and nodes), and then use the current observations to correct our predictions, which are then pushed up to correct y^t. **C.** We predict the x_i^t's using the previous observations and the *posterior* of y^t, thus passing along the shared information from the group to each individual. **D.** The filtering step for the x_i^t's uses the current observations to correct the prediction estimate for each object.

we are treating it as a parameter, we use a semicolon behind the conditioning bar.

$$p(y^t|z^t_{1..M}) \propto p(y^t|z^{t-1}_{1..M}) \int \prod_i p(z^t_i|x^t_i)p(x^t_i|y^t, x^{t-1}_i) \times p(x^{t-1}_i|z^{t-1}_i; y^{t-1})\, dx^t_{1..M}, x^{t-1}_{1..M} \quad (10)$$

Again, since we assume the value of y is observed, the x_i's become independent which allows us to update them individually. This trick allows us to bypass the costly matrix inversion operation that is the bottleneck in the exact inference algorithm. Thus, we have traded a single matrix inverse operation over a matrix that grows with the number of objects in a group, with M matrix inverse operations, each one over a much smaller matrix and with constant size. As a result, our approximate inference algorithm scales linearly with the group size. The posterior distribution for x_i is given in Eq. 11.

$$p(x^t_{1..M}|z^t_{1..M}; y^t) \propto \prod_i p(z^t_i|x^t_i)p(x^t_i|z^{t-1}_i; y^t) \quad (11)$$

Finally, we provide pseudo code for our approximate inference algorithm which makes use of the pseudo-posterior distributions given in Eq. 10 and 11. Next we discuss learning the model parameters from data.

D. Model Parameters

The parameters of *ClustKFM*, Q, W, and α, can be learned from data using expectation-maximization (EM) [13]. Note, the terms A and H are fixed according to the problem since A describes the movement (e.g. $loc^t = loc^{t-1} + vel^{t-1}$) and H describes which values from the state vector are observable (e.g. only capture position, not velocity). We use the standard KFM EM algorithm to learn the appropriate value for Q [13]. We use EM to learn W and α as well, though due to space

Algorithm 2 Approximate inference algorithm

```
1: // Group Update: $y$
2: predict $y$ according to $p(y^t|y^{t-1})$
3: for $i = 1 \to M$ do
4:     compute prior of $x_i^t$ using $x_i^{t-1}$ and predicted value of $y^t$
5:     compute $x_i^t$'s correction to $y^t$ from $z_i^t$ (and prior of $x_i^t$)
6:     $yCorrection+ = yCorrection_i$
7: end for
8: update $y^t$ from the $x_i$ corrections: $y^t = y^{t-1} + yCorrection$
9: // Individual Updates: $x_i$'s
10: for $i = 1 \to M$ do
11:     compute prior of $x_i^t$ using $x_i^{t-1}$ and posterior value of $y^t$
12:     compute $x_i^t$'s correction from $z_i^t$ (and improved prior of $x_i^t$)
13: end for
```

constraints we only briefly describe the learning procedure for α here.

The value of α, which controls how to weight the group versus the individual during filtering, is an important parameter because it controls the tightness of a group's clustering. To learn this value we employ EM and take the derivative of the log likelihood of our model set it to zero and solve for α. However, since we require that $0 \le \alpha \le 1$, we introduce a Lagrange multiplier, λ, to handle this constraint in our maximization problem. Finally, we have Eq. 12, where $v = [x_{i,t} - A(\alpha y_t + (1-\alpha)x_{i,t-1})]$.

$$\frac{\partial}{\partial\alpha} \mathcal{LL}(x, y, z|\Theta) = \frac{\partial}{\partial\alpha} [-\frac{1}{2}\sum_{t=2}^{T}\sum_{i=1}^{M} v^T Q^{-1} v + \lambda(\alpha - 1)] = 0 \quad (12)$$

We use the Kuhn-Tucker Lagrangian conditions [14] since we require that $\alpha \ge 0$ as well as restricting the maximum value to be at most 1. Solving for α and λ, we find the update equation for α's *maximization* step (we omit this due to space limitations). The *expectation* step consists of computing smoothed estimates of the belief state for each moving object, x_i, as well as the group parameter, y. Iterating between these two steps, we are able to estimate the value of α from the data. In our experiments, we typically see convergence within 20 iterations.

In addition to learning the clustering parameter, α, we must also split the objects into groups such that the movement patterns within each group is similar. Using a short sequence of observations from the set of moving objects, we identify objects that are likely to move in a similar manner. We assume that the observed sequence is representative and objects will continue to move in this manner, at least into the near future. The same data may be used to learn both the groups of moving objects and α and this process may be repeated periodically to insure *ClustKFM* maintains its accuracy. In our experiments we learned α once for the entire trajectory, however, online approaches to maintaining clusters and re-learning α are possible rather than batch re-learning.

Due to the location update uncertainty, we propose tracking each individual object with a KFM and clustering the inferred trajectory for each object to identify a good partition. When there is a large degree of location update uncertainty, directly using the noisy observations to cluster typically results in a poor partitioning of the objects since the trajectories are very inaccurate. Instead, we first compute the smoothed trajectories using a KFM for each object and use these estimates to cluster the objects. For clustering, we used k-means, setting k to 10% of the database size.

IV. Efficient Processing for Probabilistic Threshold Range Queries

In this section we take advantage of the existing group structure for answering probabilistic threshold range queries (PTRQs). Rather than placing the mobile objects into an index structure, which requires additional maintenance, we utilize the grouping aspect of our tracking model to efficiently process PTRQs. We develop a bounding region (BR) which is guaranteed to contain all objects within a group with probability of at least Q_T. That is, we limit the probability that *any* object lies outside of our BR which allows us to prune true negatives when processing PTRQs.

We focus on the PTRQ, defined below, because it is a fundamental operation for managing the locations of uncertain mobile objects [15], [16], [17]. As with other moving object databases (MODs), we assume that a user may only query the current and future positions of moving objects. We first provide an overview of how to go about bounding the probability of the location of uncertain objects, then describe what exactly we need to compute during tracking and at query time, lastly, we describe how to maintain the bounds for answering time interval queries.

Definition 1: **PTRQ**$(R, [t_s, t_e], Q_T)$**:** A PTRQ takes as input a query region, R, the start and end times, $[t_s, t_e]$, and a probability threshold, Q_T. The query returns an object, O_i, if $\exists p_i^t \geq Q_T$, for any $t \in [t_s, t_e]$, where $p_i^t = \int_R p(x_i^t.loc)dx_i^t.loc$, and $x_i^t.loc$ is the uncertain location of object O_i at time t according to the tracking model.

A. Group Bounds

Since our proposed tracking model already maintains the group structure of mobile objects with similar movement behavior, we would like to utilize this rather than placing the objects into an index structure which requires additional overhead. We exploit the existing structure to develop a bound on the uncertain locations of objects within each group, which can then be used to prune true negatives from the query result set. However, bounding the positions of uncertain objects is difficult since objects change (both position and uncertainty) over time and we need to guarantee that objects cannot satisfy the query *given a specific probability threshold.* This threshold only becomes available at query time, thus we cannot precompute the BR for each group. Instead, we specify the BR as a function of the probability threshold, such that once the query is known, we can quickly determine the size of the BR and test if it intersects with the query. The bounding region is illustrated in figure 4.

To parameterize the BR on the query probabilistic threshold, we utilize the structure of the Gaussian distribution. The Gaussian can be viewed as a function of the Mahalanobis distance from the expected value. In fact, this view is useful because it standardizes probabilities across all bivariate Gasussians. That is, the probability mass contained within a Mahalanobis distance of 1 is the same for any bivariate Gasussian, independent of the specific parameter values. However, the Euclidean extent of the ellipse required to encompass this mass may differ substantially depending on the parameters. Specifically, the (Euclidean) length of the axes of an ellipse at a Mahalanobis distance of 1, can be computed using Eq. 13.

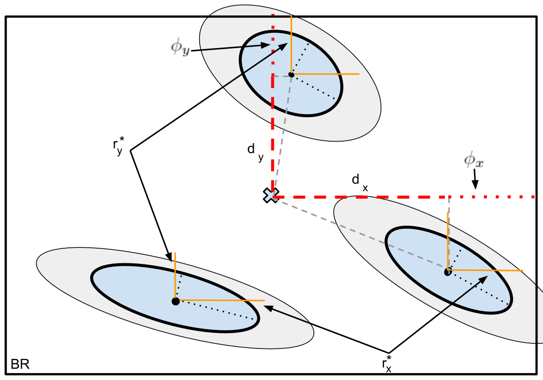


Fig. 4. An example probabilistic group bound for 3 uncertain objects. The bounded probability mass is given by the inner ellipse. The maximum distance from the mid point to the expected location of each object is labeled $d_{x/y}$. This distance is extendent by $\phi_{x/y} = max_i(r^*_{x/y}(i))$, where $r^*_{x/y}(i)$ is the extent of the bounded ellipse of object i in each dimension, considering the angle of rotation of the ellipse. Finally, the bounding region (BR) is constructed using the dimensions: $d_x + \phi_x$ by $d_y + \phi_y$.

$$r_{x/y} = \frac{1}{2}\left(\sigma_x^2 + \sigma_y^2 \pm \sqrt{(\sigma_y^2 - \sigma_x^2)^2 + 4(\rho\sigma_x\sigma_y)^2}\right) \qquad (13)$$

The standard ellipse for a Gaussain contains approximately 67% of the probability mass. Depending on the query probability threshold, we need to expand or reduce the size of this ellipse in order to capture the appropriate probability from each object. To accomplish this, we observe that the amount of probability mass of a multivariate Gaussian, contained within a certain Mahalanobis distance is described by the χ^2 distribution, $f_{\chi^2}(sc, df)$, where the parameter, sc, is the Mahalanobis distance from μ, and df is the degrees of freedom, corresponds to the data dimensionality (2 in our case). At query time, we are given a probability threshold, Q_T, which is the minimum probability mass that can fall within the query region for the uncertain object to satisfy the query. We can use this value to work backward and identify the size of the error ellipse we must bound, $P_{BR} = 1 - Q_T$, such that if the BR does not intersect the query region, then these objects may be pruned. From the definition of the χ^2 cdf, we can analytically solve for Mahalanobis distance, x, given this probability threshold, P_{BR}. The expression is given in Eq. 14,

$$\begin{aligned} F(sc;k) &= \frac{\gamma(\frac{k}{2}, \frac{sc}{2})}{\Gamma(\frac{k}{2})} = P_{BR} \\ sc &= -2log(-P_{BR} + 1) \qquad (14) \end{aligned}$$

where $\gamma()$ is the lower incomplete gamma function. Eq. 14 allows us to compute the Mahalanobis distance of the ellipse bounding the area containing the given probability of an object, P_{BR}, at query time. That is, if our BR contains this error ellipse, and does not intersect the query region, we can safely prune all of the mobile objects within the BR since the amount of probability mass that lies within the query region for any object cannot exceed Q_T. Thus far, we have described an approach for identifying an ellipse for each object which must be encompassed within the BR in order to guarantee correctness. Next, we discuss how to construct the BR given this information for each object.

To extend this idea to groups with multiple uncertain objects, we must identify the bounding region which contains all of the encompassing ellipses. We first identify the group center, g_{center}, by computing the midpoint, in each dimension, from the expected values of the objects in the group. Using, g_{center}, we can compute the maximum distance from any object to the midpoint, $d_{x/y}$. This provides us with a region that contains each of the expected values, we can expand this extent in each dimension by the maximum axis of the standard ellipses for the objects in the group, $\phi_{x/y} = max_i(r^*_{x/y}(i))$ (where r^*_x measures the extent in the x dimension considering the angle of rotation). Combining these pieces gives us our final definition for the BR as a function of Q_T.

$$r_{x/y} = \phi_{x/y} sc + d_{x/y} \qquad (15)$$

The necessary information required to compute a BR for each group can be maintained efficiently as the objects are being processed by the tracking algorithm. We compute the group center g_{center} by maintaining the minimum and maximum point in each dimension from the set of expected locations of the mobile objects within each group. Likewise, the maximum value of ϕ is maintained while the covariance matrices are being updated. The total required computation at tracking time is $O(M)$, however, since we already update the location of each object, this computation essentially comes for free.

The computation required at query time is (i) compute the Mahalanobis distance for Q_T (Eq. 14) and (ii) compute the bounding region extent (Eq. 15). The resulting computation time is $O(1)$. Next, we describe how to use the BR to prune groups from PTRQs over time intervals.

B. Processing PTRQs

Answering time slice queries is a simple matter of checking if the BR intersects the query region, R, given the probability threshold, Q_T. This can be computed in a similar manner to time slice queries in the TPR-tree [18].

Time interval queries are more difficult because we must maintain the integrity of the BR over time as the objects are moving and their location uncertainty is growing. The BR growth can be segmented into two factors: (i) the movement of the group and (ii) the growth in uncertainty of the location of each object. To maintain the tightest BR possible, we allow g_{center} to move with the group. We set the velocity in each dimension as the mid point of the minimum and maximum velocity of the objects. Because the objects can move at different velocities, the BR radius must grow to accommodate this difference, $vr_{x/y}$. Additionally, since queries are over the current and *predicted* locations of mobile objects, the location uncertainty of each object grows with time. To guarantee that our bounds maintain the specified probability mass for each object over time, we must capture this growth as well. From our model, the covariance of the predicted state of an object is computed as $\Sigma_{pred} = APA^T + Q$, where P is computed according to section III-B.1.

Intuitively, the growth in uncertainty of an object's position over time is driven by the uncertainty in the object's velocity, $\sigma_{vx/vy}$, and the correlation between velocity and position, ρ (this can be seen by writing out the matrix multiplication for the predicted covariance). Thus to accommodate this growing uncertainty over time, the BR radius must increase to cover the appropriate ellipse of each object. The covariance growth is $vc_{x/y} = \rho_{v,p}\sigma_{vx/vy}$, where $\rho_{v,p}$ is the correlation coefficient linking the velocity and position and $\sigma_{vx/vy}$ is the standard deviation of the velocity in a given dimension. Finally, the BR extent in each dimension, as a function of time is given in Eq. 16.

$$r_{x/y}(t) = (v_{x/y} \pm (vr_{x/y} + vc_{x/y}))(t) + (g_{x/y} \pm r0_{x/y}) \qquad (16)$$

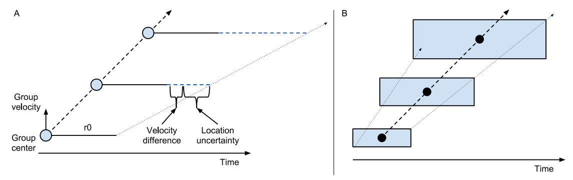


Fig. 5. An example of the evolution of a BR over time due to the two factors: (i) object movement and (ii) object location uncertainty. Figure A shows the bound growth in a single dimension over time, breaking down the different factors. Figure B shows an example of the full rectangular BR expanding over time.

Figure 5 provides an example of the group changing over time. It shows the movement of g_{center} as well as the radius expansion as time progresses.

For evaluating time interval PTRQs, we use a process similar to the TPR-tree [18]. We must check for the three basic cases: (i) the query region completely consumes the group bound, (ii) the group bound consumes the query region, or (iii) the group bound and the query region intersect at some point during the time interval. These cases are illustrated in figure 6. Because of the constraints of linear movement, each of these checks needs only be evaluated at the first and last time step of the query time interval. If any of the cases returns an overlap in both the x and y dimension, then we must evaluate the individual objects, otherwise, we can prune the entire group. We evaluate the effectiveness of this bounding approach in the experimental section on both real and synthetic datasets.

V. Experiments

We evaluate both the quality of our proposed tracking model and the efficiency of our pruning-based query processing approach. We compare the exact and approximate inference

[5]http://www.rtreeportal.org/datasets/trajectories/buses.zip

[6]The query region size parameter is defined in terms of the proportion of the total extend in each dimension

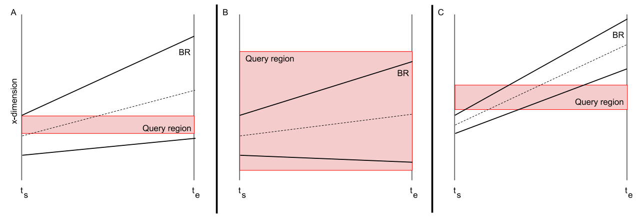


Fig. 6. Example interactions between a query region and a bounding region (BR) (note: only the x-dimension is shown here). In (A) the query region is completely contained within the BR, (B) the query region completely contains the BR, and (C) the query region intersects the BR during the time interval.

algorithms for our model (*ClustKFM* and *ApproxClustKFM*) with other common approaches to tracking moving objects. As a baseline, we assume that location updates provide the precise position for each object and simply use the update as our location estimate and compute velocity by taking the difference over consecutive updates, this method is referred to as *Naive* in our experiments. To account for the uncertainty inherent in the location updates, we also compare to a basic Kalman filter model (KFM). Since this model does not consider the known correlation between groups of moving objects, we refer to this model as the *Independent Kalman filter* (*IndKFM*). Lastly, we develop a simplistic approach which integrates the known dependency between objects into the KFM by expanding the model state space from maintaining the position of a single object to jointly modeling the positions and velocities of all moving objects. We keep the structure of the KFM, but maintain the complete joint distribution over the location and velocity of all objects in our belief state. That is, instead of maintaining a belief state of four values ($[loc_x, loc_y, vel_x, vel_y]$), it becomes a vector of length $4M$ and a covariance matrix of $4M$x$4M$ where off-diagonal values indicate the degree of covariance between each pair of objects. This method is referred to as the *Joint Kalman filter* (*JointKFM*).

TABLE II
EXPERIMENTAL PARAMETERS

Name	**Description**
Location Uncertainty	50, **100**, 250, 500
Group Size	5, **10**, 25, 50
Number of Objects	**5k**
Query Specific Parameters	
Probability Threshold	0.25, **0.5**, 0.75
Time Interval Length	2, **5**, 10
Query Region Size [6]	0.2, **0.35**, 0.5, 0.65

Table II provides an overview of our evaluation parameters. Unless otherwise stated, we use the parameters in bold in our experiments.

A. Datasets

We generate synthetic data by sampling from the generative process of *ClustKFM* which ensures that the group of generated trajectories will exhibit correlated movements. Since the other tracking models do not make any assumptions about the dynamics of the entire group of objects (except for *JointKFM*), this still provides a fair comparison to other techniques. We use synthetically generated data to test the computational scalability of the update procedure of our model as well as to test our procedure for learning the value of α in *ClustKFM*. We do not show our results examining tracking accuracy on synthetic datasets due to space limitations.

We also evaluate the ability of our model to track the movement of coordinated groups of uncertain mobile objects using two sets of real GPS traces briefly described in table III. Each record of the raw data traces contains an object identifier, latitude and longitude readings, and a timestamp. After segmenting the individual objects, we split trajectories that run for long periods into individual trajectories, each of which runs for $N = 100$ time steps. We then align the times for all of the trajectories such that the i^{th} observation for each object in the database corresponds to the same time.

TABLE III
REAL GPS TRACES

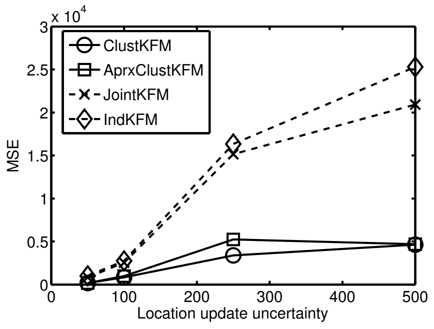
Name	**Description**
Buses [4]	A collection of GPS trajectories from 7 buses and 53 trucks collected in Athens, Greece.
TurkeyVultures [2]	GPS trajectories of 26 Turkey vultures collected during their winter migration.

Datasets are constructed by generating noisy observations about the actual trace for each object, in all of our experiments we generate 20 trials and average the results. We allow *ClustKFM* and *ApproxClustKFM* to use the first 10 observations to learn α and then evaluate the tracking accuracy over the remainder of the trajectory. We examine the tracking quality of our model, using the same inference algorithm for both learning and inference. That is, *ClustKFM* uses exact inference for learning and tracking, while *Approx ClustKFM* uses the approximate inference algorithm for both.

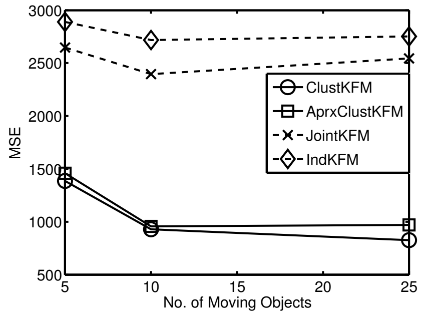
In sections V-B and V-C we evaluate the tracking quality of our models on a single cluster of coordinated uncertain moving objects to examine how the group size and location uncertainty affect the tracking model in an isolated setting. These results show the tracking accuracy for a single instantiation of *ClustKFM* tracking a single cluster. In section V-D, we examine the query efficiency and quality achieved by several instantiations of *ClustKFM* (one for each group) applied to a complete dataset.

B. Tracking Quality

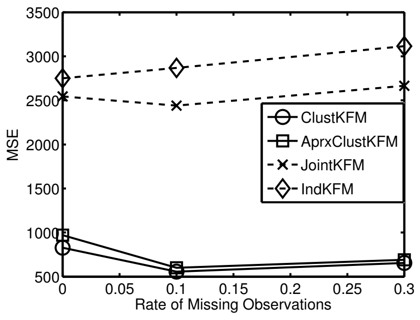
We first evaluate the accuracy of our tracking model for coordinated groups on the two real datasets by computing the mean squared error (MSE) in estimated expected location. Our experiments show that *ClustKFM* consistently outperforms the other state of the art tracking models, including those currently in use in MODs. In all of our experiments, the MSE for the naive tracking model was consistently at least an order of magnitude worse than all of the competing models and we therefore do not show it in our figures. We note, however, that the failure of the naive tracking method to capture the



(a) Location Update Uncertainty

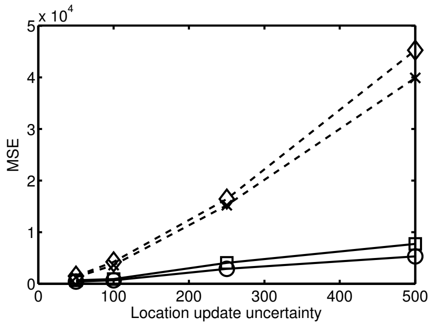


(b) Group Size

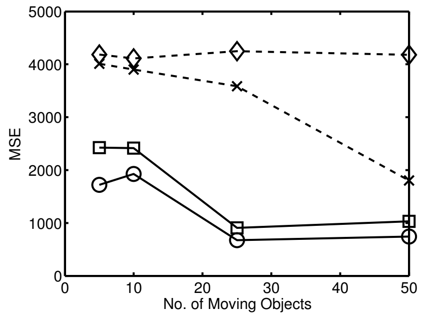


(c) Rate of Missing Location Updates

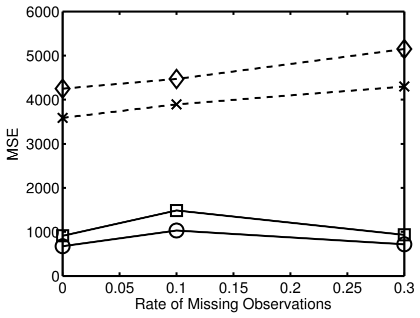
Fig. 7. Mean squared error (MSE) of the different tracking models on the TurkeyVultures dataset.



(a) Location Update Uncertainty



(b) Group Size



(c) Rate of Missing Location Updates

Fig. 8. Mean squared error (MSE) of the different tracking models on the Buses dataset.

approximate positions of mobile objects makes a strong case for managing uncertainty in the data if it exists. We compare the effect of the the update uncertainty (R), group size (M), and rate of missing (at random) location updates on the ability of each tracking model to accurately estimate the positions of the set of moving objects. Figure 7 shows the results from the TurkeyVultures dataset and figure 8, from the Buses dataset. Interestingly, *ApproxClustKFM* seems to consistently perform nearly as well as *ClustKFM*, indicating that our approximate inference algorithm sufficiently captures the group behavior.

Figure 7(a) and 8(a) show that our *ClustKFM* models are not only more accurate, but as the degree of uncertainty of the mobile objects grows, our models deteriorate at a much slower rate. In figures 7(b) and 8(b), we see that our tracking estimates improve slightly as the size of the group grows. However, this effect seems to wear off after $M = 25$. Lastly, we note that our tracking models are fairly resistant to missing updates since we are able to harness observations of other objects within the group to aid in our inferences.

C. Tracking Inference Scalability

Tracking the positions of uncertain mobile objects must be performed quickly as new updates arrive for each object, so it is important that this operation be efficient. Figure 9 shows a comparison of the update computation time as a function of the number of objects in the group for each of the different models. We see that *ApproxClustKFM* scales linearly with the number of objects. In contrast, the running time for *ClustKFM* (exact inference) and *JointKFM* both increase sharply once we have around 100 objects to track (within a single group). The complexity for these algorithms is $O(M^3)$ (where M is the size of the group) due to the required inversion of the covariance matrix. In contrast, *Approx ClustKFM* and *IndKFM* perform M inversion operations on covariance matrices of constant size, thus reducing the complexity to $O(M)$. The constant in *Approx ClustKFM* is slightly larger because the covariance matrices include the state of both the cluster state, y^t, and the object, x_i^t.

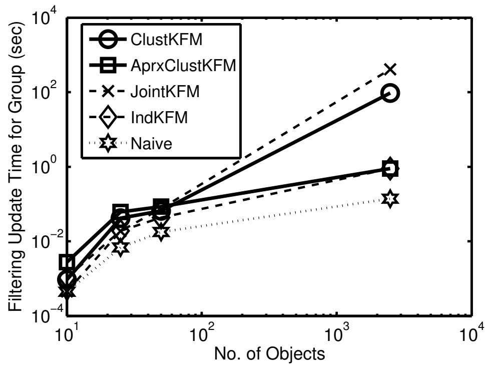


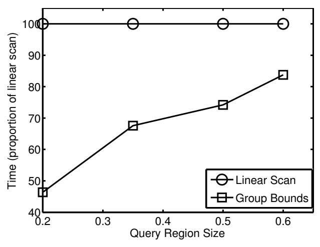
Fig. 9. Model update (inference) computation time.

D. Query Processing

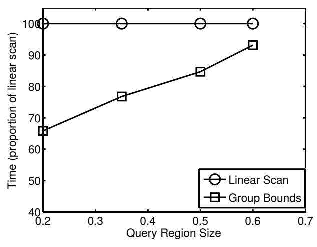
We first evaluate the effectiveness of the group bounds for pruning objects from a query on synthetically generated data traces so that we could control the various aspects of the data. We show results in terms of the proportion of time required to compute the complete linear scan over the data since this measure is robust to the specific method implemented to evaluate the probability of each object within a query region (e.g. some form of approximating a multivariate Gaussian CDF). For each dataset, we cluster the trajectories and use a separate instantiation of *ClustKFM* to track each group. For each object, we select an update uncertainty by sampling $\sigma_i \sim \mathcal{N}(100, 10)$, mimicking varying location signal quality. Queries are sampled uniformly over the space of the trajectories. For each query region size, we generate 30 trials (each with a time interval of length $|T| = 5$). We first validate our pruning on synthetically generated data under various conditions, then evaluate both querying efficiency and quality on the two real datasets.

In figure 10 we evaluate the pruning power of our group bounds on synthetic data. In general, as the size of the query region increases, we are able to prune fewer groups and thus the processing times approach that of the linear scan. This is expected since larger query regions will likely contain more objects in the result set, thus removing the possibility that we can prune these groups. In figure 10(a), we see that the query time is fairly consistent independent of the specific query probability threshold. In contrast, figure 10(b) shows that the length of the query time interval can significantly reduce our pruning capabilities. Again, this may be due to the fact that more objects end up in the result set as well as the increases in our group bounds. Additionally, figure 10(c) shows that only the highest level of uncertainty ($\sigma = 500$m) in the positions of the mobile objects tends to cause the group bounds to deteriorate significantly. Overall, the groups bounds are generally able to prune $20 - 50\%$ of the uncertain mobile objects from a query. While such an approach is not suitable for large scale MODs due to the need to track each individual mobile object, we do observe considerable gains in efficiency without the need for a traditional index structure.

Next, we query the real datasets and evaluate the processing efficiency of the group bounds. Figure 11 shows the processing time for both datasets. Similar to the experiments on the synthetic data, we are able to cut down processing time between $20 - 50\%$ for small query regions, with a linear decrease in pruning power as the query size increases. The bounds for the TurkeyVultures dataset appear to perform very well. This is likey due to the natural tightly coordinated movement of these animals and the fact that they travel freely in (i.e. they are not constrained to a road network). In contrast, the busses in the Buses dataset tend to cluster less tightly. This can be seen in the model accuracy experiments in figure 8 as well. Lastly, we evaluate the effect of the tracking model accuracy on the resulting queries. Figure 12 shows the F-measure of the query results for each of the different tracking models on both datasets. In both cases, given the degree of uncertainty in the data, *ClustKFM* and *ApproxClustKFM* provide significantly higher quality results than the competing tracking models. In the TurkeyVultures dataset, the *ClustKFM* models outperform

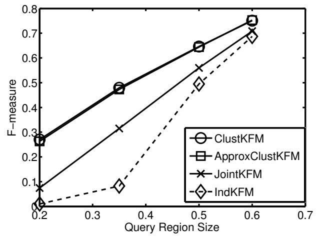


(a) Turkey Vultures

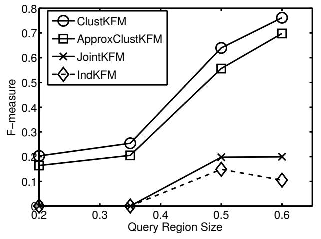


(b) Buses

Fig. 11. Efficiency of processing PTRQ using group bound pruning on real datasets.



(a) Turkey Vultures



(b) Buses

Fig. 12. F-measure for the results of the PTRQ on real datasets using the various tracking models.

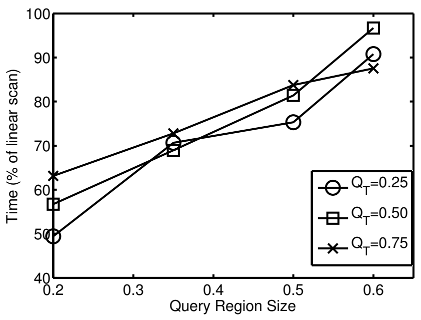
the alternative models, however, even *JointKFM* provides a substantial improvement over *IndKFM*. This provides evidence of the tightly coordinated group movement in this dataset. In contrast, we see a slight decrease in overall performance by the *ClustKFM* models in the Buses dataset which is likely due to the restricted movement and less tightly structured group movement in this data. Additionally, we notice very consistent results between *ClustKFM* and *ApproxClustKFM*, showing that our approximate inference algorithm is able to adequately utilize the coordinated group structure to improve tracking quality.

VI. Related Work

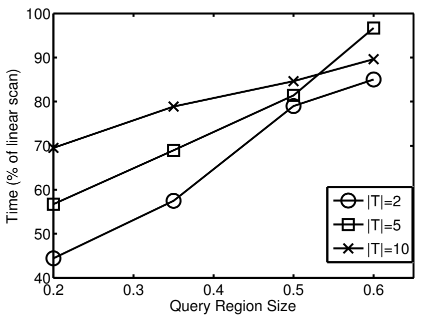
In this section, we review the relevant research in the areas of movement models and querying and indexing mobile objects. For a more comprehensive survey of these areas, see [19].

A. Movement Models

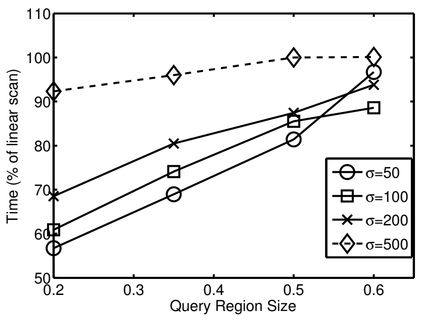
Several recent papers have introduced simple models of the movement of objects and integrated them into moving object databases (MODs) [15], [20], [17]. A model of the object movement is necessary for answering predictive queries and also provides a mechanism for answering conditional or ‘what if’ queries (e.g. *what if the object was actually at location A at time t_i?*). Although these models consider the location uncertainty of moving objects, they only consider the uncertainty introduced as a result of the time since the object’s last update. That is, after an object updates its position it could potentially change speeds or direction slightly thus making the *current* location uncertain. In addition to this ‘time-lag’ uncertainty, the position updates themselves could also



(a) Query Probability Threshold



(b) Query Time Interval Length



(c) Uncertainty

Fig. 10. Query processing time, as a percentage of the time to perform a linear scan over the entire dataset using our group bounds to prune objects without evaluating their probabilities. We show how the processing time varies (a) over different probability thresholds, (b) for varying query time intervals, and (c) for varying levels of location uncertainty in the mobile objects.

be uncertain. This is often the case as moving objects are likely to drift in and out of GPS coverage, and therefore may rely on less precise localization techniques (e.g. cellular tower triangulation) to provide position updates.

Gao and Cao [21] addresses the problem of characterizing user mobility using a discrete time Hidden Markov Model (HMM) to represent an objects motion over time. The authors describe user movements by modeling the direction changes (hidden state) and inferring transient and steady state behavior of objects. While this work focuses on predicting long term and short term movement *behavior*, we are interested in utilizing group structure to accurately track moving objects online as new uncertain location updates arrive.

In the field of robotics, researchers have used particle filtering [22], [10], [8] and Kalman filters [9], [11], [23] for decades. Particle filtering is a sequential Monte Carlo technique for estimating distributions using iterative sampling. This approach is flexible in that it can estimate arbitrary distributions, however, it suffers from the same drawbacks of other sampling techniques, namely large computation requirements. Alternatively, the Kalman filter is based on the assumption that all uncertainty is Gaussian and therefore offers more efficient processing.

Other approaches for combining dependent variables in a filtering routine have been introduced previously [22], [24], [25]. Most of this work stems from computer vision in which distributions rarely fit well to normal distributions due to various uncertainties that arise in imaging, and these methods all implement various forms of particle filters which incorporate variable dependence. In contrast, this paper introduces a model that incorporates a simple dependence structure between a large group of variables while maintaining an efficient and exact inference procedure that is suitable for tracking large groups of uncertain mobile objects.

From the statistical signal processing community, the problem of group tracking tackles a different problem in which observations must first be associated with individual objects before they can be tracked [26], [27], [28], [29]. While these methods solve a very difficult problem, they typically do not scale to more than 10 targets.

Additionally, Tasi et al. [30] consider the problem of tracking a group of mobile objects through a sensor network. The authors objective is to reduce communication costs by utilizing the similar movement pattern of the group. In contrast, our goal is to reduce uncertainty and improve tracking accuracy.

B. Moving Object Databases

Moving object databases (MODs) are database systems that maintain the current position of a set of moving objects. Objects update the MOD periodically with their latest position and the MOD uses this information to make predictions about where the object will travel next. To answer queries in between time steps, the MOD can use a model of the objects' movement [31], [15], [17], [32]. Additionally, to manage a large number of mobile objects, several index structures have been introduced which deal with the objects' frequently changing positions [33], [16], [18], [34], [35], [17].

Most similar to our work is that of Zhang et al. [17], in which the authors present a new model for moving objects that handles uncertainty in both location and velocity and then modifies the B^x-tree to index uncertain mobile objects. The uncertain moving object model proposed defines a probability distribution over location and velocity (independently) by griding a two dimensional space for both parameters and assigning a probability to each cell. Thus the model induces an extra error since probabilities are assigned to groups of parameters. Additionally, the authors introduce a B^x-tree modified to index uncertain objects by indexing each grid cell of the object's location separately.

In the area of querying uncertain trajectories, Trajcevski [5] has introduced a simple uncertainty model for spatiotemporal data in which each object is allowed to deviate from its expected trajectory by a maximum distance of δ. Objects are treated as linear piece-wise cylinders with a radius of δ. Additional work from Trajcevski et al. [6] has addressed answering continuous (time) nearest neighbor queries over uncertain trajectories using the uncertainty model proposed earlier. As mentioned in [36], this does not provide a model of the uncertain location and can therefor result in inconsistent path hypotheses.

VII. Conclusions

In this work, we address the problem of managing coordinated groups of mobile objects with location uncertainty by introducing a novel tracking model. We exploit the dependencies between individual objects within a coordinated group, resulting in more accurate location estimates which produces higher quality query results. We performed experiments on two real datasets and show that our tracking model is able to reduce the average location error by up to a factor of 4X. Additionally, we have taken advantage of the group structure imposed by our tracking model during query processing. We have developed probability bounds which enable us to prune groups of objects which cannot satisfy a PTRQ, thereby removing the need to evaluate the probability of each object individually (as in a linear scan). Our experiments show that this pruning can reduce query processing by as much as 50%. These findings corroborate our intuition that exploiting shared information among objects can significantly reduce data uncertainty and improve tracking accuracy. Moreover, our proposed model and approximate inference algorithm achieve these improvements with only minor increases in computational complexity.

Acknowledgment

The first author was partially funded by a National Science Foundation Graduate Research fellowship. Additionally, this research was sponsored in part by the U.S. National Science Foundation under grant IIS-1219254 and by the Army Research Laboratory under cooperative agreement W911NF-09-2-0053 (NS-CTA). The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, “Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet,” in *ASPLOS*, 2002, pp. 96–107.

[2] `http://www.movebank.org/`.

[3] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, “Leveraging spatio-temporal redundancy for RFID data cleansing,” in *SIGMOD*. New York, New York, USA: ACM Press, 2010.

[4] M. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, “Entracked: energy-efficient robust position tracking for mobile devices,” in *MobiSys*, 2009, pp. 221–234.

[5] G. Trajcevski, “Probabilistic range queries in moving objects databases with uncertainty,” in *MobiDE*, 2003, pp. 39–45.

[6] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. Cruz, “Continuous probabilistic nearest-neighbor queries for uncertain trajectories,” in *EDBT*, 2009, pp. 874–885.

[7] C. S. Agate and K. J. Sullivan, “Road-Constrained Target Tracking and Identification Using a Particle Filter,” in *SPIE*, no. 805, 2003.

[8] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, “Particle filters for positioning, navigation, and tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[9] R. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[10] Z. Chen, “Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond,” Adaptive Systems Lab, McMaster University, Hamilton, Ontario, Tech. Rep., 2003.

[11] G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” University of North Carolina at Chapel Hill, Tech. Rep., 2006.

[12] R. D. Shachter and C. R. Kenley, “Gaussian Influence Diagrams,” *Management Science*, vol. 35, no. 5, pp. 527–550, 1989.

[13] Z. Ghahramani and G. E. Hinton, “Parameter estimation for linear dynamical systems,” University of Toronto, Technical Report CRG-TR-96-2, Tech. Rep., 1996.

[14] D. Klein, “Lagrange multipliers without permanent scarring,” http://www.ee.columbia.edu/ vittorio/LagrangeMultipliers-Klein.pdf.

[15] B. S. E. Chung, W.-C. Lee, and A. L. P. Chen, “Processing probabilistic spatio-temporal range queries over moving objects with uncertainty,” in *EDBT*, 2009.

[16] J. Hosbond, S. Saltenis, and R. Ortoft, “Indexing uncertainty of continuously moving objects,” in *International Workshop on Database and Expert Systems Applications*. IEEE, 2003, pp. 911–915.

[17] M. Zhang, S. Chen, and C. S. Jensen, “Effectively Indexing Uncertain Moving Objects for Predictive Queries,” in *VLDB*, 2009.

[18] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, “Indexing the positions of continuously moving objects,” in *SIGMOD*, June 2000, pp. 331–342.

[19] N. D. Larusso and A. Singh, “Sensing for mobile objects,” in *MANAGING AND MINING SENSOR DATA*. Springer, 2013, ch. 10.

[20] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu, “Prediction and indexing of moving objects with unknown motion patterns,” in *SIGMOD*. ACM Press, 2004.

[21] W. Gao and G. Cao, “Fine-Grained Mobility Characterization: Steady and Transient State Behaviors,” in *MobiHoc*, 2010.

[22] E. Koller-Meier and F. Ade, “Tracking multiple objects using the condensation algorithm,” *Robotics and Autonomous Systems*, vol. 34, no. 2-3, pp. 93–105, 2001.

[23] J. Yim, J. Joo, and C. Park, “A kalman filter updating method for the indoor moving object database,” *Expert Systems with Applications*, 2011.

[24] F. Moreno-Noguer, A. Sanfeliu, and D. Samaras, “Integration of Dependent Bayesian Filters for Robust Tracking,” *ICRA*, pp. 4081–4087, 2006.

[25] D. Schulz, W. Burgard, D. Fox, and A. Cremers, “Tracking multiple moving objects with a mobile robot,” in *CVPR*, vol. 1. IEEE, 2001, pp. I–371.

[26] A. Carmi, F. Septier, and S. Godsill, “The gaussian mixture mcmc particle algorithm for dynamic cluster tracking,” in *Information Fusion, 2009. FUSION’09. 12th International Conference on*. IEEE, 2009, pp. 1179–1186.

[27] A. Gning, L. Mihaylova, S. Maskell, S. Pang, and S. Godsill, “Group object structure and state estimation with evolving networks and monte carlo methods,” *Signal Processing, IEEE Transactions on*, vol. 59, no. 4, pp. 1383–1396, 2011.

[28] R. Mahler, “Multitarget bayes filtering via first-order multitarget moments,” *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 39, no. 4, pp. 1152–1178, 2003.

[29] S. K. Pang, J. Li, and S. J. Godsill, “Detection and Tracking of Coordinated Groups,” *IEEE Transactions On Aerospace And Electronic Systems*, vol. 47, no. 1, 2009.

[30] H.-p. Tsai, D.-n. Yang, and M.-s. Chen, “Mining Group Movement Patterns for Tracking Moving Objects Efficiently,” *Transactions on Knowledge and Data Engineering*, vol. 23, no. 2, pp. 266–281, 2011.

[31] R. Cheng, D. Kalashnikov, and S. Prabhakar, “Querying imprecise data in moving object environments,” *TKDE*, vol. 16, no. 9, pp. 1112–1127, 2004.

[32] K. Zheng, G. Trajcevski, X. Zhou, and P. Scheuermann, “Probabilistic range queries for uncertain trajectories on road networks,” in *EDBT*, 2011, pp. 283–294.

[33] S. Chen, C. S. Jensen, and D. Lin, “A Benchmark for Evaluating Moving Object Indexes,” in *PVLDB*, 2008, pp. 1574–1585.

[34] Z. Song and N. Roussopoulos, “Seb-tree: An approach to index continuously moving objects,” in *Mobile Data Management*. Springer, 2003, pp. 340–344.

[35] Y. Tao, D. Papadias, and J. Sun, “The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries,” in *VLDB*, 2003.

[36] T. Emrich, H. Kriegel, N. Mamoulis, M. Renz, and A. Züfle, “Querying uncertain spatio-temporal data,” in *International Conference on Data Engineering (ICDE)*. IEEE, 2012, pp. 354–365.