A Constructive Bayesian Approach for Vehicle Monitoring

Y. Xiang and V. Lesser
140 Governor's Drive, Department of Computer Science, University of Massachusetts
Amherst, MA 01003-4610, USA E-mail: {yxiang, lesser}@cs.umass.edu

Abstract

A key component of a vehicle monitoring system is uncertainty management. Bayesian networks (BN) emerged as a normative and effective formalism for uncertain reasoning in many AI tasks. Since a priori modeling of the domain into a BN is impractical due to the vast interpretation space, the BN formalism has been considered inapplicable to this type of task. We propose a framework in which the BN formalism can be applied to vehicle monitoring. The framework explores domain decomposition, model separation, model approximation, model compilation and re-analysis. Experimental implementation demonstrated good performance at near-realtime.

Keywords: Bayes net, sensor fusion, tracking.

1 Introduction

Vehicle monitoring (also known as *tracking*) takes as input the measurements from a surveillance region which is populated by a number of moving objects (vehicles), and estimates the number of vehicles as well as their type and movement. Measurements entering into a vehicle monitoring system are the output of a signal processing system which directly processes the sensor output. Uncertainty involved in the task includes the unknown number and types of vehicles, the unknown association of measurements and vehicles, the inaccuracy of measurements, potential missing measurements, environmental noise, and "ghost" measurements.

Traditional engineering approach [2; 1] applies Kalman filtering, which normally requires linearity and Gaussian assumptions in modeling. Traditional AI approach [3] is based on incremental vehicle track construction and ad-hoc measures of uncertainty.

Bayesian probability theory has been applied to uncertain reasoning in many AI problems in the form of Bayesian networks (BNs) [10; 8]. The formalism allows representation of uncertain dependence relations that go beyond linearity and Gaussian assumptions. For most problems tackled, a domain model in the form of a BN is constructed before observations are available and inference takes place. For vehicle monitoring, due to the unknown number of vehicles and the almost infinite number of track patterns by multiple vehicles, construction of a BN model a priori is impractical. This difficulty has led to the issue whether the BN formalism is applicable to vehicle monitoring type of problems [3]. Recently, the formalism has been applied to guide automated highway vehicles [6] and to identify individual vehicles appearing on highway surveillance cameras [7], although the issues addressed are different from those in our task, which focuses on identifying vehicle tracks in open regions.

We explore several general ideas: decomposition of the problem into quasi-independent subproblems, approximation in modeling to reduce complexity, model compilation to speed up runtime computation, and focused re-analysis for error reduction. We show that by exploring these ideas, the BN formalism can be applied to vehicle monitoring, and our experiment with randomly simulated vehicle scenarios (fig 1) showed good performance.

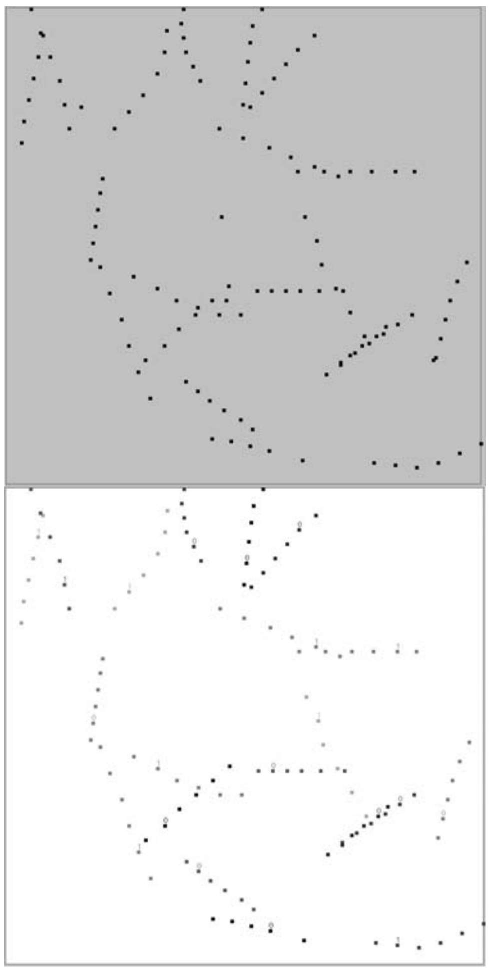
Figure 1: Top: A scene of 20 vehicles over 6 time instants. Bottom: Interpretation. Tracks differ in color with each labeled by a vehicle type code.

2 Bayesian formulation

We consider the measurements obtained from an open surveillance region at k discrete instants $t = 1, ..., k$. We assume $k > 3$ so that accelerations of maneuvering vehicles can be extracted. Denote the set of measurements at $t = i$ by $D_i = \{d_{ij} | j = 1, ..., m_i\}$. The total set of measurements is then $D = \{D_i | i = 1, ...k\}$, which we refer to as a *scene*. Each measurement is either produced by a vehicle of a particular type or is due to noise. Noisy measurements may be unrelated to any vehicles, or may correspond to vehicle movement as in the case of a "ghost".

A *full trajectory* is a set of k measurements $r = \{d_{1j_1}, ..., d_{kj_k}\}$. A *partial* trajectory is a *proper* subset of a full trajectory. If all measurements in r are produced by the movement of a vehicle w and no other measurement in D are also produced by w, then r is the *track* of w. We assume[1] that there are no vehicles entering and leaving the region between $t = 1$ and $t = k$. Hence when there are no missing measurements, each vehicle *track* is a unique full trajectory from D. Otherwise, each track is a unique full or partial trajectory from D. A *ghost* track is similarly defined. Two vehicles may be very closely located at time t so that they are perceived by the sensors as a single measurement. Without losing generality, we regard the measurement as being generated by one of them and regard the measurement at t as missing for the other vehicle.

An *interpretation* T of D is a partition of D into a set Y of full or partial trajectories and a set N of measurements. Each trajectory in Y represents a *believed* track and measurements in N represent believed noise unrelated to any tracks.

The task is then to find T such that $P(T|D)$ is maximal among all interpretations, where $P(T|D)$ reads "the probability of T being the interpretation of D". This task corresponds to the *track formation* in the tracking literature as opposed to *track maintenance* where each new measurement is to be associated with an already established track.

In the literature some researchers (e.g., [9]) assume multiple measurements for a vehicle at each time instant while others (e.g., [2]) assume a single measurement. In this work, we assume a single measurement for a vehicle at each time instant. Such restriction does not compromise the generality as multiple measurements can usually be grouped by their closeness and summerized as a single measurement.

Figure 1 (top) shows a simulated scene of 20 vehicles with $k = 6$. The total number of measurements is 123. Some vehicles have missing measurements, e.g., the track at the middle bottom of the scene. Environmental noise is present in the scene. An easily identifiable one is at about the center of the scene. A less obvious one is near the lower end of the track at the right edge of the scene. Figure 1 (bottom) shows the interpretation with the highest $P(T|D)$ (See Section 10), where each identified track is drawn with a different color (shown at a different gray level) and noise has been identified and removed.

3 Direct method

A *direct* method would be to compute $P(T|D)$ for each T and then choose the one with the maximal value. An interpretation T is *feasible* if for every pair of trajectories r and s in T, $r \cap s = \emptyset$. Otherwise, T is *infeasible*. If T is infeasible, then $P(T|D) = 0$. A feasibility test hence rules out interpretations with certainty.

[1]How to address the cases where the assumption does not hold is discussed in Section 9.

How do we go about computing $P(T|D)$ for a feasible interpretation T? We can model the problem in a fashion of hypothesis-causes-features: If T is the correct interpretation, then each trajectory in T must behave like a track. Using Bayesian networks as a representation of probabilistic causal models, it suggests the following: Create a binary hypothesis variable $\mathbf{T} \in \{true, false\}$ with the semantics "T is the correct interpretation of D". For each trajectory r in T, create a binary child variable $\mathbf{r}$ of $\mathbf{T}$ with the semantics "r represents a track".

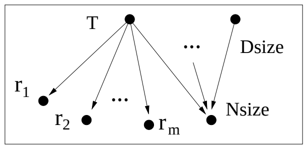

Figure 2: Interpretation model as a Bayesian net.

Furthermore, the set N of measurements must behave like noise. How can this be represented in a Bayesian network? We may think of N as (1) not supporting any trajectories that behave like tracks, and (2) occupying an expected portion of the total measurements. The behavior (2) can be represented as a discrete child variable **N size** (cardinality of N) conditioned on some parents. One obvious parent is **T**. Other parents may include **D size** (cardinality of D) or other parameters that may affect the noise model. The structure of a BN thus constructed is shown in fig 2. Note that the structure is *interpretation specific*. The number of children of **T** varies for each interpretation. The behavior (1) does not seem to lend itself to an explicit representation. We claim that it has been encoded in the above structure implicitly. Consider interpretations T_1 and T_2, which are identical except a trajectory r_1 in T_1 is entirely contained in N_2 of T_2. Suppose r_1 behaves well like a track and N_1 has a well expected proportion of D based on expected frequency of noisy data. Using the above representation, T_2 will have one less positive support (r_1) for being a correct interpretation and one additional negative support (N_2 out of portion). Consequently, probabilistic inference using the two corresponding BNs will result in $P(T_1|D) > P(T_2|D)$.

To summerize, $P(T|D)$ can be computed using the BN in fig 2, which we refer to as the *interpretation model*.

4 Modeling trajectory

In the BN of fig 2, variables $\mathbf{r_i}$ are not directly observable. Hence each of these variables must be elaborated with a *trajectory model*. Each measurement contains the measured location of a vehicle at a given time. It may also contain the energy level of the measurement, the frequency range (in the case of passive sensing of acoustic signals), and other relevant feature information. We refer to the corresponding components of trajectory model as *movement* model, *frequency* model, and so on.

First, we consider the location information in the measurements (the movement model). To simplify discussion, we restrict it to 2D locations. Denote the location of a vehicle by (x, y). Denote the magnitude and angle of velocity vector by v and ω, and the magnitude and angle of acceleration vector by a and η. The movement of a vehicle can be represented as a dynamic Bayesian network in fig 3 (left). The upper layers of each slice models the acceleration (a, η), velocity (v, ω) and location (x,y) of the vehicle at a particular time instant. Arcs from a slice to the next models how the state of the vehicle depend on the previous state. The location of vehicle is not directly observable but through the potentially inaccurate measurements. This is modeled by the measurements (x', y') which are dependent on the true location (x, y) as well as the measurement error e. In the model, measurement errors are assumed to be independent, but correlated errors can also be modeled (with increased inference computation cost).

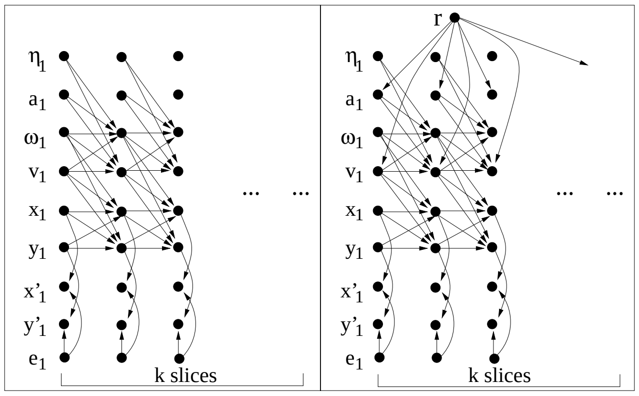

Figure 3: Vehicle model (left) and trajectory movement model (right).

A trajectory may or may not correspond to a true vehicle track. We convert the above vehicle model into a trajectory movement model by adding the root variable $\mathbf{r}$, which models whether the trajectory behaves as a vehicle track. We make $\mathbf{r}$ the parent of each variable a and variable v. The conditional probability distribution $P(a|\mathbf{r} = true)$ models the acceleration of a vehicle. The distribution $P(a|\mathbf{r} = false)$ models an arbitrarily generated trajectory. We do not model the angles ω and η as the children of $\mathbf{r}$ due to the the following assumptions:

We assume that each vehicle of a particular type can only move in a given range of v values. It can however move at any directions. Within the v range, at any time it may freely choose acceleration value a in another range with no restriction on the angle η. Strictly speaking, the freedom on η is an approximation as vehicles may have different acceleration ranges for tangential directions and lateral directions. However, the approximation helps to simplify our model and it seems to be quite reasonable: A running car cannot make a very sharp turn, but neither can it speed up or slow down abruptly. We allow total freedom in η values in our simulated vehicles (see fig 1) and they do not seem unrealistic.

Given the assumptions, the values of ω and η provide no differentiating power between a track and a non-track, and hence are not dependent on $\mathbf{r}$ in our model.

In addition to the movement, other information contained in the measurements can also help evaluate if a trajectory behaves like a track. For example, measurements corresponding to a true track may have similar energy levels and closely related signal frequencies. For each feature at each time instant, a child variable of $\mathbf{r}$ can be created in the trajectory model in fig 3 (right) if these features are independent when they are produced by the same vehicle.

In principle, given an interpretation of m trajectories, we can complete the interpretation model in fig 2 by extending each $\mathbf{r_i}$ node with a trajectory model. Then belief propagation can be used to compute $P(T|D)$.

5 Decomposition of scene into islands

The direct method discussed in Sections 3 and 4 require the explicit evaluation of all interpretations. Unfortunately, it is intractable even for a scene of a few tracks.

Consider a scene with $k = 6$ where 4 measurements per time instant are obtained. The total number of full trajectories is $4^6 = 4096$. The total number of partial trajectories with one missing measurement is $4^5 * 6 = 6144$. Hence the total number of trajectories with one possible missing measurement is 10240. To find the most probable interpretation, a total of 2^{10240} interpretations need be evaluated. Note that although many of these interpretations are infeasible, a feasibility test (Section 3) has to be explicitly performed for each.

Our first basic idea for making the computation tractable is to decompose the problem into independent or semi-independent subproblems which are easier to solve. In particular, we decompose a scene into smaller independent or semi-independent groups of measurements. Only interpretations within a single group are explicitly evaluated, while interpretations across multiple groups are ignored as much as possible.

We apply two levels of decomposition. The first level decomposes a scene into independent groups which we refer to as *islands* defined below. The second level is presented in Section 6. Given two (location) measurements d and e, $|d - e|$ denotes the distance between them. Let $MAXD$ denote the maximum distance any vehicle may travel in one time interval plus twice the maximum location error.

Definition 1 *An* `island` *in a scene is a subset L of measurements such that for each $l \in L$ and each $d \in D \setminus L$, $|l - d| > MAXD$.*

The decomposition is only quadratic on $|D|$, but the computational savings by using islands can be tremendous. Consider the previous scene of 24 measurements. If the scene can be decomposed into two islands with 2 measurements per time instant per island as shown in fig 4, then for each island the total number of full and partial trajectories with one missing measurement is $64 + 192 = 256$ (512 for the scene), a significant reduction from the previous 10240.

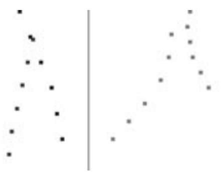

Figure 4: A scene of 4 tracks decomposed into two islands (divided by the straight line).

What will be the error introduced by island decomposition? If there is no missing measurements in the scene, then every trajectory corresponding to a true track is contained in a unique island. Hence island decomposition introduces *no* error at all. In fact, use of islands introduces no error even when limited missing measurements are present as formalized in the following proposition.

Proposition 2 *In a scene with at most missing measurements at $t = 1$ or $t = k$, an exhaustive evaluation based on islands yields the identical result as one without using islands.*

Proof:

Since no measurements are missing at $t \in \{2, ..., k-1\}$, each true track is contained in one island and will be evaluated.

On the other hand, without using islands, each interpretation T containing trajectories crossing multiple islands will be evaluated. For each such trajectory r, $P(r\ is\ a\ track|D)$ will be very low due to impossible velocity/acceleration values. This in turn will produce very low $P(T|D)$, resulting rejection of T as the final interpretation. $\square$

When measurements are missing at $t \in \{2, ..., k-1\}$, a track with one measurement missing may be split into two islands and not be evaluated at all. Let the probability of a missing measurement be q. The probability that an isolated track is split in the middle due to one missing measurement is $(k-2)q$. Although its value increases with k, we assume that k is a small integer in track formation. Measurements obtained after k time instants will either be used one instant at a time (as in track maintenance) or processed as additional k length scenes. For $k = 6$ and $q = 0.02$, we have $(k-2)q = 0.08$.

In fact, the above estimation is a very conservative upper bound. The threshold $MAXD$ is determined by the fastest possible vehicles to be expected. For a slower vehicle, the distance traveled in two time intervals may still be less than $MAXD$ and hence the missing measurement does not cause the track to be separated into two islands. Furthermore, when multiple tracks are present, two sections of a broken track may be included in an island if other tracks are close enough to both. In Section 9, we discuss how to further reduce the error under island decomposition due to missing measurements.

Assuming that each island can be independently interpreted, we obtain

$$\begin{aligned} & P(T|D) = P(T_1, T_2, \ldots, T_m | L_1, L_2, \ldots, L_m) \\ = \quad & P(T_1|T_2, \ldots, T_m, L_1, \ldots, L_m) \ldots P(T_m|L_1, \ldots, L_m) \\ = \quad & P(T_1|L_1) \ldots P(T_m|L_m) \end{aligned}$$

where each L_i is an island and T_i is the interpretation of L_i. Hence, we only need to find interpretation T_i for each island such that $P(T_i|L_i)$ is maximal. We will then have $T = \cup_i T_i$. Algorithm 1 outlines the top level control of our scene interpretation system.

Algorithm 1 (Scene interpretation)

```
 Input: A scene $D$.
decompose $D$ into islands;
for each island $L_i$ containing at least one trajectory
    process $L_i$ to get the island interpretation $T_i$;
    add $T_i$ to the scene interpretation $T$;
return $T$;
```

6 Decomposing islands to peninsulas

Although islands are easier to deal with than the original scene, due to possible track crossing, near-parallel tracks, or other types of adjacency, a large island may still contain measurements of several tracks. When this is the case, the combination explosion illustrated earlier occurs again at the island level. We apply a second level of decomposition within each *large* island to make the evaluation of large islands more manageable:

Definition 3 *A* `peninsula` *is a subset S of measurements in an island L such that the following conditions hold:*

1. *For time $t = 1$, S has exactly one measurement d_1 called* `initiator`.
2. *For each $t \geq 2$, S contains each $d_t \in L$ such that there exists $d_{t-1} \in S$ and $|d_t - d_{t-1}| < MAXD$.*

Intuitively, if the initiator of a peninsula belongs to a track, then the entire track is contained in the peninsula. As an example, consider an island made of two tracks that are nowhere close except at time $t = k$ ($k = 6$). Based on the previous calculation, 2^{256} interpretations should be evaluated. The island produces two peninsulas and each contains only k measurements. Hence the total number of full trajectories and partial trajectories with one missing measurement in each peninsula is $1+6 = 7$, and the total number of interpretations to be evaluated for the island becomes $2(2^7) = 256$. Although this represents the best scenario, in general, whenever the starting segment (t is close to 1) of a track is "clear" (no nearby measurements from other tracks at the same time frames), decomposition into peninsulas will reduce the number of interpretations to be evaluated.

We may extend the definition of peninsula to allow the initiator to be a measurement at time $t = k$. The corresponding peninsula is then a *backward* peninsula (versus the *forward* peninsula as defined above).

What error might be introduced by using peninsula? When there are no missing measurements, each track is contained in at least one peninsula and will be evaluated. Hence, evaluation using peninsula introduces no error at all. However, error may occur when missing measurements are present. Consider the island shown in fig 5 (a). It contains measurements from two tracks, one of which is drawn in squares and the other in ovals. The time of each measurement is also shown. The upper track has the measurement at $t = 2$ missing. The two forward peninsulas found are shown in (a) as rounded areas. The two backward peninsulas are shown in (b). None of the peninsulas contains all measurements of the upper track. Hence this track will not be evaluated.

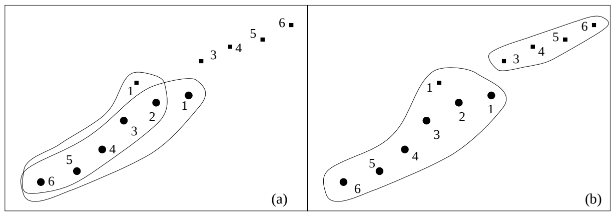

Figure 5: (a) Forward peninsulas in an island. (b) Backward peninsulas in the island.

The following proposition identifies an error-free condition when using peninsulas.

Proposition 4 *If an island only has missing measurements at $t = 1$ or $t = k$, each track is either contained in a forward peninsula or a backward one.*

Proof:

Let r be a track with measurements $\{d_1, ..., d_k\} \setminus \{d_i\}$ ($i = 1$ or $i = k$). If $i = 1$, all measurements are contained in the backward peninsula with initiator d_k. If $i = k$, all measurements are contained in the forward peninsula with initiator d_1. □

Proposition 4 suggests that we may generate both forward and backward peninsulas for all measurements at $t = 1$ and $t = k$. Evaluation using these peninsulas is resistant to at least a percentage of $2/k$ of errors due to one missing measurement in a track. The value $2/k$ is a lower bound because a track with a missing measurement at t such that $1 < t < k$ may still be contained in a peninsula due to the presence of measurements from other tracks in the same island.

Our decomposition using islands and peninsulas can be *equivalently* formulated using an adjacency graph where there is a link from a point to another if the distance in between is less than $MAXD$. Whether to maintain and search such a graph explicitly or implicitly is a design choice.

7 Model separation

After using peninsulas to generate trajectories for an island, we can evaluate each interpretation T (we overload the notation T here for the island) using a completed interpretation model (Sections 3 and 4). Since a trajectory may participate in multiple interpretations, this method will duplicate evaluation of a given trajectory multiple times.

To reuse the evaluation of each trajectory, we evaluate T using a set of BNs: a top level BN as in Section 3 and one trajectory BN for each trajectory as in Section 4. The evaluation of each trajectory BN is performed separately. After evaluation of each trajectory in T is completed, the results are used in the evaluation of the top level BN to produce $P(T|D)$. The evaluation result of a trajectory r can then be reused for the evaluation of each interpretation that r participates.

Evaluation computation can be performed in several ways. We briefly describe the cluster tree method [8]. The method groups variables in a BN into overlapping subsets called *clusters*. The clusters are organized into a tree. Probabilistic inference is performed by message passing (belief propagation) along the tree. With one round of *inward* propagation towards an arbitrary cluster followed by another round of *outward* propagation away from the cluster, the updated probability for each variable can then be obtained in any cluster containing it. More details can be found in the above reference.

Since each trajectory BN shares a single variable $\mathbf{r_i}$ with the top level BN, if we convert each BN into a cluster tree and join each trajectory tree with the top level tree at the cluster containing $\mathbf{r_i}$, the resultant cluster tree is equivalent to that created without model separation (fig 6). Since we are only interested in the posterior distribution on variable $\mathbf{T}$ which is contained in the top level tree, belief propagation consists of only *inward* propagation toward a cluster containing $\mathbf{T}$ (as shown by arrows in fig 6).

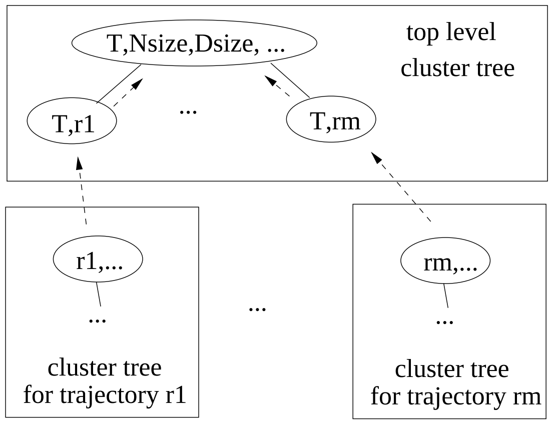

Figure 6: Belief propagation in cluster trees. Each oval represents a cluster. The tree on the top is converted from Figure 2.

Most trajectories in a scene are not due to actual tracks and will receive very low evaluation. The separation of trajectory evaluation and interpretation evaluation also allows those trajectories to be eliminated so that the interpretations they participate in are effectively discarded without being explicitly evaluated.

An additional advantage of model separation is that it allows variables shared by different models to be represented at the right degree of coarseness at each model. Each variable $\mathbf{r}$ is shared by the interpretation model and a trajectory model. In the trajectory model, $\mathbf{r}$ can be given the domain $\{not\_track, type0\_track, type1\_track, ...\}$. The differentiation of vehicle types is not only an interesting result, but also facilitates model building.

On the other hand, in the interpretation model, it is sufficient for $\mathbf{r}$ to convey only whether the corresponding trajectory is a track. The type of the vehicle is not important. In fact, having to differentiate vehicle types in the interpretation model will be an unnecessary burden in model building. Separation of model evaluation allows each $\mathbf{r}$ in the interpretation model to be represented as a binary variable. During inference, the trajectory model can sum its posterior distribution first before feeding to the interpretation model.

Based on island decomposition and model separation, the island level control of our scene interpretation system is outlined in Algorithm 2.

Algorithm 2 (Island interpretation)

```
 Input: An island L.
if L is not too large
    evaluate each trajectory;
    get interpretation T of L from highly evaluated trajectories;
else
    decompose L into peninsulas;
    for each peninsula S
        evaluate each trajectory;
    get interpretation T of L from highly evaluated trajectories;
return T;
```

The interpretation generation/evaluation is outlined in Algorithm 3.

Algorithm 3 (Interpretation evaluation)

```
 Input: A set R of trajectories.
for each interpretation T from R
    perform feasibility test on T;
    if T passed
        construct interpretation BN model;
        compute $P(T|D)$ using the BN;
        if $P(\mathbf{T} = correct|D)$ is the highest so far, store T;
return stored interpretation;
```

8 Movement model reduction

Each feasible trajectory in each peninsula can be evaluated using the trajectory model (Section 4). $P(\mathbf{r}|\mathbf{D})$ can be computed using any one of several common inference algorithms (see [4] for a recent survey). We consider the complexity using the cluster tree method [8]. For $k = 6$, a good cluster tree has about 31 clusters. About one third of them each has a size of 7 variables. If the domain size for acceleration (a), velocity (v), location (x, y) and measurement (x', y') is at least 10, then the belief state space of many clusters will be huge. Even if the inference computation is affordable, when it must be repeated for each of hundreds or more of feasible trajectories, it is very expensive and near real-time monitoring becomes impossible. Although a query DAG [5] can be used to speed up the inference, its complexity is comparable to the original algorithm used to generate the query DAG. Hence, a query DAG does not provide the magnitude of computational savings needed.

Instead, we explore the following alternative: since we are primarily interested in $P(\mathbf{r})$, we try to reduce the model such that only $\mathbf{r}$ and observables are left. However, using x' and y' as observables will end up with a model where every variable is strongly dependent on every other. The cluster tree of the model will have a cluster of huge state space. The alternative is to use observed velocity/acceleration. Each observed velocity is computed using two adjacent location measurements and each acceleration is computed using three as follows (assuming unit time interval):

$$v'_1 = \sqrt{(x'_2 - x'_1)^2 + (y'_2 - y'_1)^2}$$

$$a'_1 = \sqrt{(x'_3 - 2x'_2 + x'_1)^2 + (y'_3 - 2y'_2 + y'_1)^2}$$

After replacing location and measurement variables (x, y, x', y'), we have the reduced model in fig 7 (a).

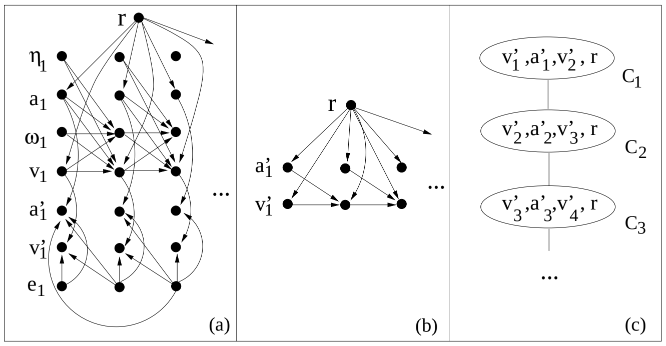

Figure 7: (a) Replacing location measurements in movement model. (b) Approximate movement model. (c) Clique chain for movement model evaluation.

Each observed velocity is dependent on two measurement errors and each observed acceleration is dependent on three measurement errors. Due to this dependence, a_1 and a_3 are not independent given $\mathbf{r}, \omega_2, v_2, a_2$ (using d-separation to (a)). However, if the value of velocity and acceleration ($|v|$ and $|a|$) are large enough than the value of measurement error ($|e|$), this dependence is not strong. If we ignore this dependence, then we obtain the Markov property: a_1 and a_3 are independent given $\mathbf{r}, \omega_2, v_2, a_2$, and v_1 and v_3 are independent given $\mathbf{r}, \omega_2, v_2, a_2$. By approximating the true value of velocity/acceleration with the observed value (effectively clumping v with v' and a with a') and removing other unobservables, we obtain the model in fig 7 (b). From (b), we obtain the cluster chain (c) which can be evaluated efficiently as derived below:

Conceptually, we follow the cluster tree method [8]. After initialization, the cluster C_1 is associated with the distribution table $P(v'_1, a'_1, v'_2, \mathbf{r})$. Other clusters have its table similarly assigned. After observations $v'_i = \nu_i$, $a'_j = \alpha_j$ $(i, j = 1, 2, ..., k)$ are obtained, they are entered into the corresponding cluster belief tables. For example, the table with C_1 becomes $P(\mathbf{r}, v'_1, a'_1, v'_2|\nu_1, \alpha_1, \nu_2)$. For each observation, instead of entering it to *one* cluster as normally performed [8], we enter into *every* cluster table that contains the corresponding variable. For example, ν_2 will be entered into tables in both C_1 and C_2.

To compute $P(\mathbf{r}|\nu_1, ..., \nu_{k-1}, \alpha_1, ..., \alpha_{k-2})$, we perform belief propagation from cluster C_1 downwards. The message from C_1 to C_2 is

$$[\sum_{v'_1, a'_1} P(\mathbf{r}, v'_1, a'_1, v'_2|\nu_1, \alpha_1, \nu_2)]/P(\mathbf{r}, v'_2).$$

This is a distribution over $\mathbf{r}$ and v'_2. At C_2, its local table is updated into the product with the message

$$\frac{\sum_{v'_1, a'_1} P(\mathbf{r}, v'_1, a'_1, v'_2 | \nu_1, \alpha_1, \nu_2)}{P(\mathbf{r}, v'_2)} P(\mathbf{r}, v'_2, a'_2, v'_3 | \nu_2, \alpha_2, \nu_3).$$

Since ν_2 has been entered into C_2, the message from C_1 to C_2 could be just

$$[\sum_{v'_1, a'_1, v'_2} P(\mathbf{r}, v'_1, a'_1, v'_2 | \nu_1, \alpha_1, \nu_2)] / \sum_{v'_2} P(\mathbf{r}, v'_2 | \nu_2),$$

which is a distribution over $\mathbf{r}$ only. That is, we have

$$\frac{\sum_{v'_1, a'_1} P(\mathbf{r}, v'_1, a'_1, v'_2 | \nu_1, \alpha_1, \nu_2)}{P(\mathbf{r}, v'_2)} P(\mathbf{r}, v'_2, a'_2, v'_3 | \nu_2, \alpha_2, \nu_3)$$

$$= \frac{\sum_{v'_1, a'_1, v'_2} P(\mathbf{r}, v'_1, a'_1, v'_2 | \nu_1, \alpha_1, \nu_2)}{\sum_{v'_2} P(\mathbf{r}, v'_2 | \nu_2)} P(\mathbf{r}, v'_2, a'_2, v'_3 | \nu_2, \alpha_2, \nu_3).$$

Note that if we had not entered ν_2 into C_2, the above equality would not hold. Second, since v'_2 has a large domain size (we used 12 in our experiments) while the new message is a distribution over $\mathbf{r}$ only, the size of the message from C_1 to C_2 is reduced significantly and correspondingly the amount of computation associated with the message passing.

Finally, we observe that

$$\sum_{v'_1, a'_1, v'_2} P(\mathbf{r}, v'_1, a'_1, v'_2 | \nu_1, \alpha_1, \nu_2) = c \; P(\mathbf{r} | \nu_1, \alpha_1, \nu_2)$$

and $\sum_{v'_2} P(\mathbf{r}, v'_2 | \nu_2) = d \; P(\mathbf{r} | \nu_i)$,

where c and d are normalizing constants. Hence we have the following efficient algorithm for computing $P(\mathbf{r} | \nu_1, ..., \nu_{k-1}, \alpha_1, ..., \alpha_{k-2})$:

Algorithm 4 (Trajectory evaluation by movement)

```
Input: $\nu_1, \ldots, \nu_{k-1}, \alpha_1, \ldots, \alpha_{k-2}$ of a full trajectory.
$B(\mathbf{r}) = P(\mathbf{r} | \nu_1, \alpha_1, \nu_2)$
for $i = 2$ to $k - 2$
    $B(\mathbf{r}) = B(\mathbf{r}) P(\mathbf{r} | \nu_i, \alpha_i, \nu_{i+1}) / P(\mathbf{r} | \nu_i)$
normalize $B(\mathbf{r})$ to get $P(\mathbf{r} | \nu_1, \ldots, \nu_{k-1}, \alpha_1, \ldots, \alpha_{k-2})$
return $P(\mathbf{r} | \nu_1, \ldots, \nu_{k-1}, \alpha_1, \ldots, \alpha_{k-2})$
```

Using this algorithm, it is no longer necessary for the on-line inference computation to actually maintain the cluster chain. This contributes significantly to realtime or near-realtime evaluation as a large number of evaluations must be performed. To obtain the parameters required by Algorithm 4, we off-line compute $P(\mathbf{r} | v'_i, a'_i, v'_{i+1})$ and $P(\mathbf{r} | v'_{i+1})$ using the accurate model in fig 7 (a). For our experiment (reported in Section 10), the off-line computation took about 12 hours using a SUN Ultra60.

Note that Algorithm 4 can be easily extended to include processing of other observations (e.g., frequency). It can also be easily modified to evaluate partial trajectories. The extension and modification are straightforward and we omit the details.

9 Re-analysis

In Algorithm 2, the operation "evaluate each trajectory" was performed for each small island and each peninsula in a large island. The operation can be performed to evaluate every full and partial trajectory. Normally, there are more partial trajectories than full ones (see examples in earlier sections). When there are no missing measurements, processing of partial trajectories is completely wasted. Even when they are infrequent, most of the processing on partial trajectories is still wasted. To achieve near real-time scene interpretation, it is desirable to reduce such processing as much as possible.

To this end, we explore *re-analysis* in the following way: For each small island and each peninsula, we only evaluate full trajectories initially. We then select highly evaluated trajectories and get the best possible interpretation T for the island L. If $P(T|L)$ is not satisfactory measured by some predetermined threshold, then the trajectory evaluation is considered inadequate and partial trajectories are evaluated before a second round of interpretation evaluation is performed.

As an example, consider Figure 5. If we search for peninsulas as defined in Definition 3 (effectively assuming no missing measurement at $1 < t < k$), a mistake will be made since the four measurements in the upper track will be considered as noise (as they are not qualified as a partial track). This will enlarge the noise set N to an unexpected level, which in turn lowers $P(T|L)$ for the best interpretation obtained. The low $P(T|L)$ will trigger a re-analysis looking for peninsulas with a missing measurement, which will identify the partial trajectory.

The *re-analysis* can be applied to a more general context: Due to the intractability of an exhaustive analysis, as we perform a bottom-up analysis (e.g., from trajectory to island to scene), we only analyze according to the most likely cases initially (e.g., the full trajectories) to make the analysis tractable. As we move up the abstraction levels, we watch for signs of failure of early analysis (e.g., the low $P(T|L)$ above) since the reality may happen to be one of those unlikely cases. When such signs are identified, we go back to a lower abstraction level, re-analyze more thoroughly and go up the abstraction levels again. Such re-analysis allows the initial analysis to be performed efficiently and allows mistakes made to be corrected with limited and focused additional computation.

In Section 2, we assumed that no entering/leaving vehicles between $t = 1$ and $t = k$. These vehicles produce tracks that are partial trajectories, some of which can already be interpreted correctly. However, if such a trajectory is too much shorter than a full one, it is likely to be interpreted as noise. Using re-analysis, the corresponding measurements can be combined with the previous or next scene to allow correct interpretation.

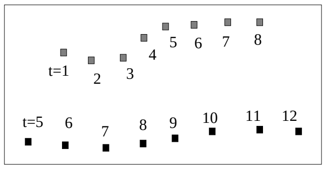

Figure 8: Two tracks across two scenes.

Consider the two tracks in fig 8 which span two scenes: s with $t = 1, ..., 6$ and s' with $t = 7, ..., 12$. The upper track corresponds to a vehicle stopping in s' and the lower track corresponds to a vehicle starting in s. The measurements from the upper track at $t = 7, 8$ are likely interpreted as noise for s'. Using the upper track r in s (from $t = 1$ to 6) as expectation, these measurements can be re-analyzed with focused processing. The measurements from the lower track at $t = 5, 6$ in s can be similarly re-analyzed using the lower track r' in s' as expectation.

10 Experimental results

To test the framework, we implemented a *scene simulator* and a *scene interpretor*. The simulator generates randomly a scene on a 200x200 grid region[2] as input to the interpretor. The interpretor is evaluated by comparing its interpretation with the simulated tracks. Each measurement contains a 2D location plus a signal frequency as would appear in passive sensing. The simulator allows us to specify the size of the region, the number of tracks in a scene, the velocity/acceleration distribution of each type of vehicles, the amount of measurements due to environment noise, and the chance of missing measurements.

A total of 280 scenes of different difficulty were simulated. The scenes were divided equally into 14 batches. Scenes in the same batch has identical size (number of tracks). The larger the size, the higher the density of measurements and the more difficult to interpret the scene. Each track consists of at most $k = 6$ measurements. Hence the total number of measurements per scene ranges from about 30 to 108 plus measurements due to noise (about 8% on average) and minus missing measurements (each measurement may be missing with a 0.02 probability).

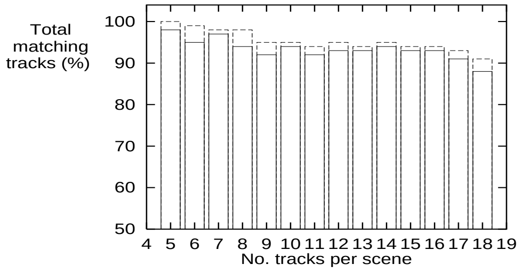

Figure 9: Summary of experimental results.

The implementation is in Java and the experiment was run using jdk1.1.8 in a Pentium II 400 under Window98. No additional runtime optimization was applied. For scenes up to 15 tracks, the average CPU time for each scene is less than 7 sec. Hence near-realtime performance was obtained for a wide range of scene sizes. For scenes with 16, 17 and 18 tracks, the CPU time are 14, 37, 46 sec, respectively, as very larger islands are frequently detected in the scenes.

An interpreted track $\mathbf{r}'$ *fully matches* a simulated track $\mathbf{r}$ (which may have missing measurements) if $\mathbf{r}'$ matches each measurement in $\mathbf{r}$. An interpreted track $\mathbf{r}'$ *partially matches* a simulated track $\mathbf{r}$ if $\mathbf{r}'$ matches each measurement in $\mathbf{r}$ except one. Figure 10 shows the percentage of fully and partially (stacked on the top) matching tracks in each batch. As the number of tracks per scene increases from 5 to 18, the percentage decreases gradually from 100% to 91%. The errors may be further reduced by combining interpretations of successive scenes, which we discuss in a longer paper.

[2]Note that the size of the grid is insignificant to the performance, but the density of the vehicles is.

Acknowledgements

This work is supported by Research Grant OGP0155425 from NSERC of Canada, by NSF under Grant No. IIS-9812755, and by DARPA and Air Force Research Lab under F30602-99-2-0525. The work is conducted while the first author is on sabbatical from Univ. of Regina.

References

[1] Y. Bar-Shalom, editor. *Multitarget Multisensor Tracking: Advanced Applications.* Artech House, 1990.

[2] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association.* Academic, 1988.

[3] N. Carver and V. Lesser. A new framework for sensor interpretation: planning to resolve sources of uncertainty. In *Proc. National Conference on Artificial Intelligence*, pages 724–731, 1991.

[4] B. D'Ambrosio. Inference in Bayesian networks. *AI Magazine*, 20(2):21–36, 1999.

[5] A. Darwiche and G. Provan. Query DAGs: a practical paradigm for implementing belief-network inference. In E. Horvitz and F. Jensen, editors, *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 203–210, Portland, Oregon, 1996.

[6] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The batmobile: towards a bayesian automated taxi. In *Proc. Fourteenth International Joint Conf. on Artificial Intelligence*, pages 1878–1885, Montreal, Canada, 1995.

[7] T. Huang and S. Russell. Object identification: a Bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103:77–93, 1999.

[8] F.V. Jensen. *An introduction to Bayesian networks.* UCL Press, 1996.

[9] V.R. Lesser and D.D. Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, 1983.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.