A Constructive Graphical Model Approach For Knowledge-Based Systems: A Vehicle Monitoring Case Study

Y. Xiang* and V. Lesser†
*University of Guelph, Canada
†University of Massachusetts, USA

July 21, 2003

Abstract

Graphical models have been widely applied to uncertain reasoning in knowledge based systems. For many problems tackled, a single graphical model is constructed before individual cases are presented and the model is used to reason about each new case. In this work, we consider a class of problems whose solution requires inference over a very large number of models which are impractical to construct a priori. We conduct a case study in the domain of vehicle monitoring and then generalize the approach taken. We show that the previously held negative belief on the applicability of graphical models to such problems is unjustified. We propose a set of techniques based on domain decomposition, model separation, model approximation, model compilation, and re-analysis to meet the computational challenges imposed by the combinatorial explosion. Experimental results on vehicle monitoring demonstrated good performance at near-real-time.

Keywords: Graphical models, Bayesian networks, dynamic model construction, probabilistic reasoning, vehicle monitoring, multitarget tracking.

1 Introduction

Graphical models [43, 41, 30, 9, 10], most notably Bayesian networks (BNs), have been widely applied to uncertain reasoning in knowledge based systems such as for medical diagnosis [26, 48], equipment troubleshooting [27, 47], financial forecasting [1], software debugging [7], etc. For many problems tackled, a graphical domain model is constructed before individual cases are presented. The model is used for reasoning about each new case. For instance, a BN for diagnosis in a medical domain is often constructed and the same model is used for diagnosis of each patient case. Alternatively, to adapt to the variation in variables and their dependence relations that are needed in modeling each case, a graphical model can be dynamically constructed after a new case is presented [6, 22, 28, 24, 44, 35, 40, 20]. In this work, we consider a class of problems which requires probabilistic inference using a very large number of dynamically constructed models. To make the study concrete, we conduct a case study in the problem domain of vehicle monitoring and then generalize the approach taken.

Vehicle monitoring (also known as *tracking*) takes as input the measurements from a surveillance region which is populated by a number of moving objects (referred to as *vehicles*)

and estimates the number of vehicles as well as their types and movements. 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, potentially missing measurements, environmental noise, and "ghost" measurements.

For vehicle monitoring, due to the unknown number of vehicles and the intractable number of track patterns by multiple vehicles, construction of a graphical model a priori is impractical. Even after a set of measurements in a given scenario has been presented, it is impractical to construct one graphical model to accomplish the reasoning task. This difficulty has led to the issue whether the graphical models are applicable to vehicle monitoring types of problems [8]. Some researchers have applied graphical models to guide automated highway vehicles [18] and to identify individual vehicles appearing on highway surveillance cameras [29]. Their problems, however, are different from vehicle monitoring, which focuses on identifying vehicle tracks in open regions.

In this research, we investigate whether graphical models can be applied to such problems where neither direct a priori modeling of the problem domain nor a posteriori modeling using a single model is practical due to combinatorial explosion. 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, graphical models can be applied successfully to the reasoning tasks in such problems. We have implemented a prototype system and tested on simulated vehicle scenarios (Figure 1) with good performance.

2 Bayesian Formulation of Vehicle Monitoring

We consider the measurements obtained from a 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 $D_i = \{d_{ij} | j = 1, 2, ...\}$. The total set of measurements is then $D = \cup_{i=1}^{k} D_i$, 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 movements as in the case of a "ghost".

A *full trajectory* is a set of k measurements $r = \{d_{1j_1}, ..., d_{kj_k}\}$, where each measurement d_{ij} is from a distinct instant. On the other hand, a *partial* trajectory is a set of $k' < k$ measurements where each measurement is from a distinct instant. If all measurements in a trajectory r are produced by movements of a vehicle w, and no other measurements 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* corresponds to a unique full trajectory from D. Otherwise, each track corresponds to 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 treat the measurement as being generated by one of them, and treat the measurement at t from the other vehicle

[1] Section 9 discusses how to handle the cases where the assumption does not hold.

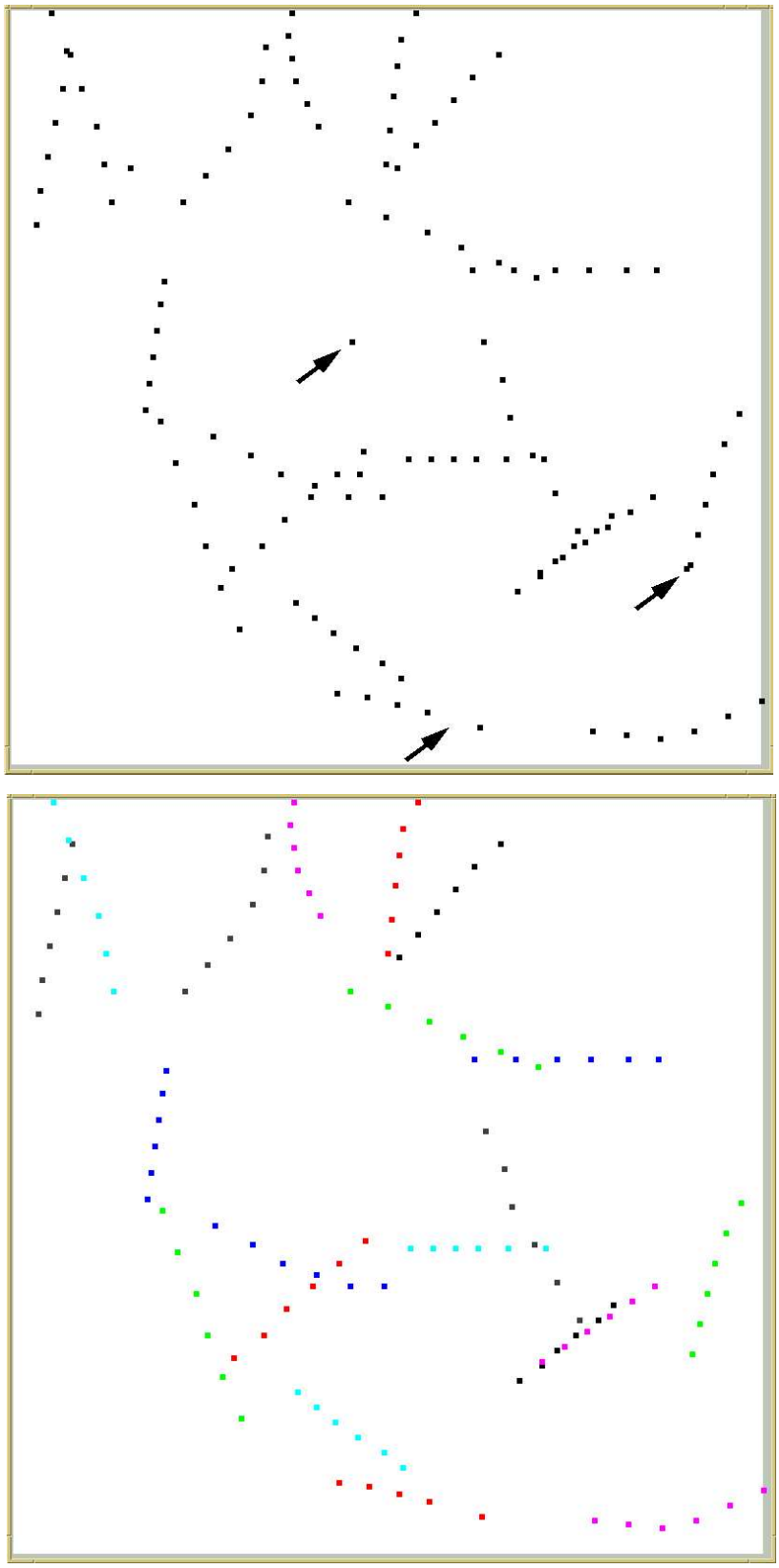

Figure 1: Top: A simulated scene with 20 vehicles over 6 time instants. Bottom: The interpretation obtained, where each identified track is shown by a different color.

as *missing*. Using the above convention, no two tracks will share any measurements. That is, for every pair of tracks r and s from D, we have $r \cap s = \emptyset$.

An *interpretation* T of D is a partition $\{r_1, r_2, ..., r_m, N\}$ of D into full or partial trajectories $r_1, r_2, ..., r_m$ and a set N of measurements. Each trajectory r_i corresponds to a *believed* track. Measurements in N correspond to 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 that T is the correct 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. Note that we have overloaded T to represent both an interpretation and the proposition "the interpretation is correct". Which meaning the symbol assumes should be clear from the context.

In the literature some researchers (e.g., [37, 45]) assume multiple measurements for a vehicle at each time instant while others (e.g., [3]) 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 bottom of the scene as indicated by an arrow. Environmental noise is present in the scene. An easily identifiable one is at approximately the center of the scene, as indicated by an arrow. A less obvious one is at the right-hand side of the scene., also shown with an arrow. Figure 1 (bottom) shows the interpretation with the highest $P(T|D)$ (See Section 11), where each identified track is drawn with a different color and noise has been identified and removed.

3 Direct Method

A *direct* method would be to compute $P(T|D)$ for each T and return the one with the maximal value. To compute $P(T|D)$ for a given interpretation T, we analyze the problem in a fashion of hypothesis-causes-features and use Bayesian networks as the causal model: If T is the correct interpretation, then each trajectory in T must behave like a track. A binary hypothesis variable $\mathbf{T} \in \{true, false\}$ is created with the semantics "T is the correct interpretation of D". For each trajectory r in T, a binary child variable $\mathbf{r} \in \{true, false\}$ of $\mathbf{T}$ is created with the semantics "r represents a correct track". Note that the two new variables $\mathbf{T}$ and $\mathbf{r}$ just introduced are in bold face, to be distinguishable from T and r.

Furthermore, the set N of measurements (see Section 2) must behave like noise. That is, N should occupy an expected portion of the total measurements. For instance, if it is expected that 10% of measurements are due to noise and an interpretation has N containing 30% of measurements, then it is highly likely that the interpretation is incorrect. This effect can be represented as a variable $\mathbf{N}$ **size** (cardinality of N) conditioned on some relevant variables. One obvious parent is $\mathbf{T}$. Other parents include $\mathbf{D}$ **size** (cardinality of D) or additional parameters that affect the noise model. The structure of a BN thus constructed is shown in Figure 2. Note that the structure is *interpretation specific*. The number of children of $\mathbf{T}$ varies for each interpretation.

The noise model works well in differentiating interpretations. Consider interpretations T_1 and T_2, which are otherwise identical except that a trajectory r_1 in T_1 is entirely contained in the noise set N_2 of T_2. Suppose r_1 behaves well like a track and N_1 occupys a proportion

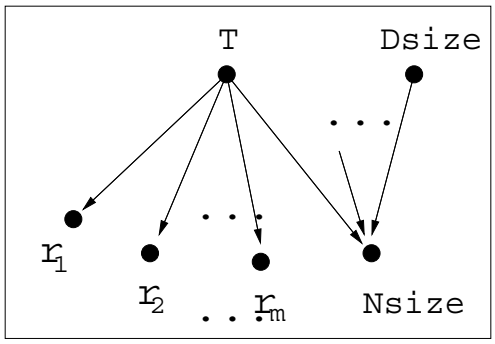

Figure 2: Interpretation model as a Bayesian network.

of D that is fairly consistent with the expected frequency of noisy measurements. 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 $P(T_1|D) > P(T_2|D)$. We use the BN in Figure 2 to compute $P(T|D)$ and refer to the BN as the *interpretation model*.

To specify the model, we need to specify the prior distribution $P(T)$. If there is no prior knowledge that allows one to prefer some interpretations over others, then an identical distribution $P(T)$ should be used for all interpretations. On the other hand, if one has some prior knowledge that favors certain interpretations, different $P(T)$ may be used for different interpretations to reflect this knowledge. In either case, the exact numerical parameters assigned are not as critical since we aim at the interpretation that maximizes $P(T|D)$.

The distribution $P(\mathbf{r}|\mathbf{T})$ can be specifies as follows: The probability

$$P(\mathbf{r} = correct\ track|\mathbf{T} = correct\ interpretation)$$

has the value close to 1. This means that a trajectory in a correct interpretation most likely behaves like a track. However, occasionally, there may be a track that moves in an unusual way. The probability

$$P(\mathbf{r} = correct\ track|\mathbf{T} = incorrect\ interpretation)$$

has the value close to 0. This means that most trajectories in an incorrect interpretation are not tracks. It is possible, however, that some trajectories can be exceptions.

4 Modeling Trajectory

In the BN of Figure 2, variables $\mathbf{r_i}$ are not directly observable. Hence, each of these variables must be elaborated with a *trajectory model*. Each measurement signifies the location of a potential 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 focus on 2-D movements. Denote the location of a vehicle at a given time by (x, y). Denote the magnitude and angle of the velocity vector by v and ω, and the magnitude and angle of the acceleration vector by a and η. Movements of a vehicle

can be represented as a dynamic Bayesian network in Figure 3 (left). Similar representation was used by Forbes et al. [18]. The upper layers of each slice model the acceleration (a, η), velocity (v, ω) and location (x,y) of the vehicle at a particular time instant. Arcs from one slice to the next model how the state of the vehicle depends on the previous state. The location of the vehicle is not directly observable and can only be inferred through the 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.

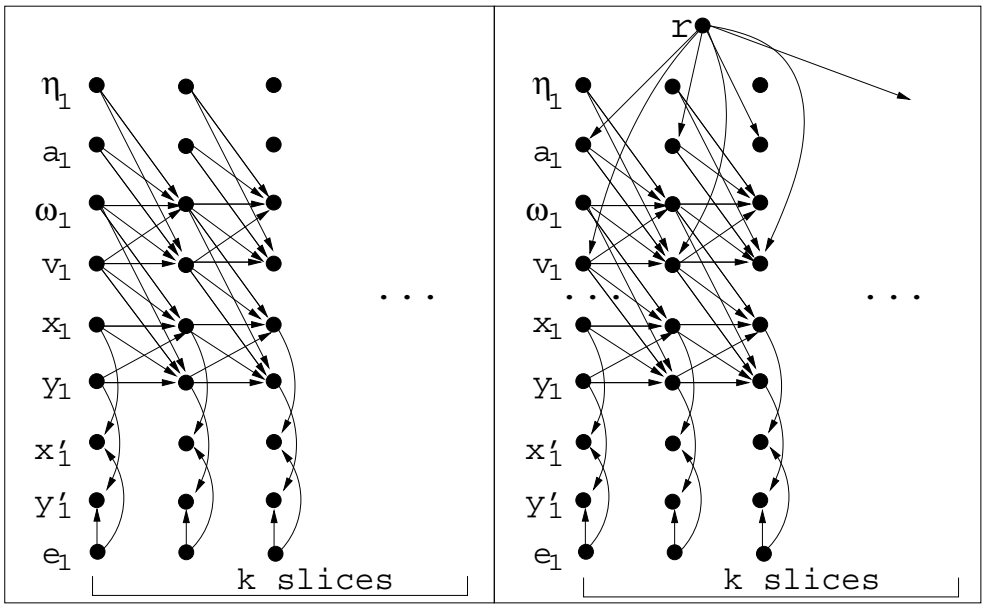

Figure 3: Left: A vehicle model as a dynamic BN, where a, η are magnitude and angle of acceleration, v, ω are magnitude and angle of velocity, x, y are location of vehicle, x', y' are location measurements, and e is the measurement error. Right: A trajectory movement model.

A trajectory may or may not correspond to a true vehicle track. We extend the vehicle movement model into a trajectory movement model by adding the root variable $\mathbf{r}$, which models whether a given trajectory behaves as a vehicle track. Note that $\mathbf{r}$ corresponds to a $\mathbf{r}_i$ variable in the interpretation model (Figure 2). It represents the proposition "the trajectory is a 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 arbitrary collection of measurements which does not correspond to a track and hence has an arbitrary value for a. Due to the the following assumptions, we do not model the angles ω and η as the children of $\mathbf{r}$:

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 Figure 1) and they do not seem unrealistic. Under these assumptions, the values of ω and η are not causally dependent on $\mathbf{r}$ in the model. Since $\mathbf{r}$

represents a hypothesis of our primary concern, this independence assumption implies that ω and η do not provide direct evidential support to confirm or reject the hypothesis. This leads to a further approximation in the run-time representation to allow more efficient run-time computation as will be seen in Section 8.

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 (in the case of passive sensing). For each feature at each time instant, a child variable of $\mathbf{r}$ can be created in the trajectory model in Figure 3 (right).

In principle, given an interpretation of m trajectories, we can complete the interpretation model in Figure 2 by extending each node $\mathbf{r_i}$ 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 requires explicit evaluation of all interpretations. Unfortunately, it is intractable even for a scene produced by a few vehicles.

Consider a scene with $k = 6$, where 4 measurements per time instant are obtained. This corresponds to a scene from 4 vehicles when there are no noise and missing measurements. 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. From these trajectories, a total of 2^{10240} combinations can be constructed. Many of these combinations are not valid interpretations. This happens when two or more trajectories in the same combination share measurements (see Section 2). Hence, a *validity test* must be performed explicitly for each combination to remove the invalid from further computation.

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 each 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 (1) for each pair $l_1, l_n \in L$ there exists a sequence of measurements $l_2, l_3, ..., l_{n-1}$ such that $|l_i - l_{i+1}| \leq MAXD$ $(1 \leq i < n)$, and (2) for each $l \in L$ and each $d \in D \setminus L$, $|l - d| > MAXD$.*

Algorithm 1 decomposes a given set M of measurements into two sets L and R such that L is an island.

Algorithm 1 (GetIsland)
Input: A set M of measurements.

$R = M$;
pick an element m from R;
$L = \{m\}$, $R = R \setminus \{m\}$;
do
  for each measurement m' in R,
    if there exists $m \in L$ such that $|m - m'] < MAXD$,
      $L = L \cup \{m'\}$, $R = R \setminus \{m'\}$;
while L is modified in the last iteration;
return L and R;

Algorithm 2 decomposes a scene D into a set of islands.

Algorithm 2 (GetAllIsland)
Input: A scene D of measurements.

$M = D$, $Islands = \{\}$;
while $M \neq \{\}$, do
  run $GetIsland(M)$ to obtain L and R;
  $Islands = Islands \cup \{L\}$;
  $M = R$;
return $Islands$;

The decomposition computation is only quadratic on $|D|$ (the cardinality of 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 Figure 4, then for each island the total number of full trajectories and partial trajectories with one missing measurement is $64 + 192 = 256$ (512 for the scene), a significant reduction from the previous 10240.

Essentially, the decomposition of a scene into islands is based on the assumption that measurements contained in an island are *sufficient* for the interpretation of the corresponding trajectories and *not necessary* for the interpretation of measurements outside the island. In other words, the interpretation of measurements that are inside the island can be performed independently of the interpretation of measurements outside the island. The computation of islands mimics *clustering*, commonly used in data analysis, but is used here for a very different purpose. Ideas similar to ours have also been explored in [16] for goal planning.

What will be the error introduced by island decomposition? If there are 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.*

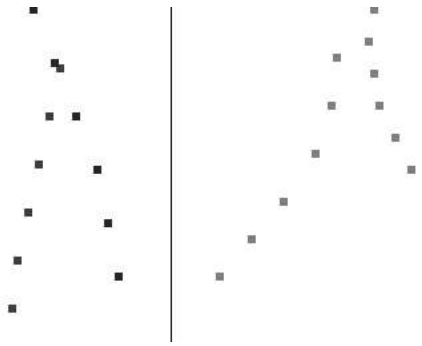

Figure 4: A scene of 4 tracks decomposed into two islands (divided by the straight line).

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, a direct method evaluates explicitly each interpretation T which contains trajectories crossing multiple islands. For each such trajectory r, $P(r \; is \; a \; track|D)$ is very small because r implies impossible velocity and acceleration values. This in turn produces very small $P(T|D)$ and hence T will not be accepted as the final interpretation. □

When measurements are missing at $t \in \{2, ..., k-1\}$, a track with one measurement missing may be split into two islands and may not be evaluated at all. Let q denote the probability that a measurement may be missing. Assume that measurements miss independently. The probability that an isolated track is split in the middle due to one missing measurement is $(k-2)\; q\; (1-q)^{k-1}$. For $k = 6$ and $q = 0.02$, this probability is 0.0723.

The above estimation corresponds to the worst case. The threshold $MAXD$ is determined by the fastest vehicles under consideration. 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, if the track with a missing measurement is no in isolation, the two sections of the broken track may still 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

$$\begin{aligned} P(T|D) &= P(T_1, T_2, \ldots, T_m|L_1, L_2, \ldots, L_m) \\ &= P(T_1|T_2, \ldots, T_m, L_1, \ldots, L_m)P(T_2|T_3, \ldots, T_m, L_1, \ldots, L_m) \ldots P(T_m|L_1, \ldots, L_m) \\ &= P(T_1|L_1) \ldots P(T_m|L_m) \end{aligned} \quad (1)$$

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 3 outlines the top level control of our scene interpretation system.

Algorithm 3 (InterpretScene)
Input: A scene D.

run GetAllIsland(D) to 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 Decomposition of Islands into 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 include measurements from several tracks. When this is the case, the combination explosion illustrated earlier occurs at the island level. We apply a second level of decomposition within each *large* island to make its evaluation 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 every $d_t \in L$ such that there exists $d_{t-1} \in S$ and $|d_t - d_{t-1}| < MAXD$.*

Algorithm 4 shows how to compute a peninsula given an initiator in an island. Note that d_t refers to a measurement obtained at time t.

Algorithm 4 (GetPeninsula)
Input: An island L of measurements and an initiator d_1.

initialize peninsula $S = \{d_1\}$;
$L = L \setminus \{d_1\}$;
for $t = 1, 2, ...,$
for each measurement d_t in L,
if there exists $d_{t+1} \in L$ such that $|d_{t+1} - d_t] < MAXD$,
$S = S \cup \{d_{t+1}\}$, $L = L \setminus \{d_{t+1}\}$;
return S;

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 $t = k$ ($k = 6$). As discussed in Section 5, 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 reduces the number of interpretations to be evaluated.

The definition of peninsula can be extended to allow the measurement at time $t = k$ to be the initiator. 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 Figure 5 (a). It contains measurements from two tracks, one of which is drawn in squares and the other in ovals. The timestamp 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.

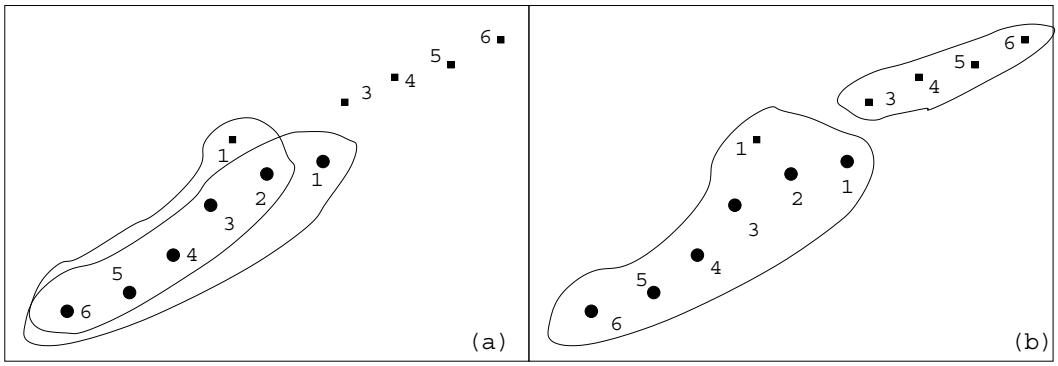

Figure 5: (a) Forward peninsulas in an island. (b) Backward peninsulas in the island.

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 the generation of both forward and backward peninsulas for all measurements at $t = 1$ and $t = k$. Although this method cannot avoid errors due to a missing measurement at t ($1 < t < k$), such errors do not always occur because the corresponding track may still be included in a peninsula due to the presence of measurements from other tracks in the same island.

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 an implementation choice.

We emphasize that decompositions of a scene to islands and peninsulas are guided by the general direction to explore subproblem independence to the largest degree possible. Such independence, once identified, can significantly reduce the computational complexity of interpretation by avoiding combinatory explosion due to trajectory combinations.

7 Model Separation

After using peninsulas to generate trajectories from an island, we can evaluate each interpretation T (we overload the notation T here for the island) with one interpretation model (Sections 3 and 4). Because a trajectory participates in multiple interpretations, this method will evaluate the trajectory multiple times.

To avoid such unnecessary re-evaluation, 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 is reused for evaluating each interpretation in which r participates.

Evaluation computation can be performed in several ways. We briefly describe the method using an alternative graphical model, called a cluster tree [30]. 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. A message is simply a probability distribution over the intersection a pair of clusters adjacent on 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 compile 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 identical to what would be created without using model separation (Figure 6). Since we are only interested in the posterior distribution on

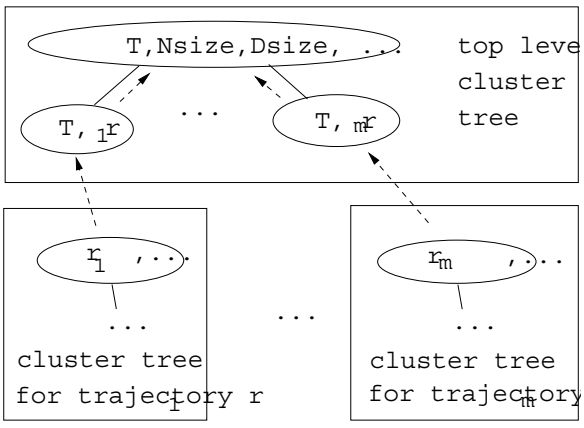

Figure 6: Belief propagation in cluster trees. Each oval represents a cluster. The tree on the top is converted from Figure 2.

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 Figure 6).

The model separation has a number of additional advantages. Most trajectories in a scene do not correspond to actual tracks and hence their probabilities will be very low. Separation of trajectory evaluation and interpretation evaluation allows those trajectories to be eliminated early so that the interpretations they participate in are effectively discarded without explicit evaluation.

Model separation also allows variables shared by different models to be represented at different degrees of coarseness as appropriate. Each variable $\mathbf{r}$ is shared by the interpretation model and a trajectory model. In the trajectory model, the domain of $\mathbf{r}$ can be $\{not\_track, type\_0\_track, type\_1\_track, ...\}$. Such refinement facilitates assessment of conditional probability distributions in model building. Evaluation of the model can also provide useful differentiation of vehicle types.

On the other hand, in the interpretation model, it is sufficient that $\mathbf{r}$ carries the belief whether the corresponding trajectory is a track. The type of the vehicle is not relevant. Hence, $\mathbf{r}$ can be a binary variable in the interpretation model. To allow the two representation coexist, before the trajectory model passes belief to the interpretation model during inference, it modifies its distribution accordingly with a simple summation:

$$P(\mathbf{r} = is\_track|D) = \sum_i P(\mathbf{r} = type\_i\_track|D).$$

Based on island decomposition and model separation, the island level control of the scene interpretation system is outlined in Algorithm 5.

Algorithm 5 (InterpretIsland)
Input: An island L.

if L is not too large,
  evaluate each trajectory;
  get interpretation T of L from trajectories receiving high evaluations;
else
  decompose L into peninsulas;
  for each peninsula S,
    evaluate each trajectory;
  get interpretation T of L from trajectories receiving high evaluations;
return T;

The last step in the *if* segment and the last step in the *else* segment are detailed in Algorithm 6. In the algorithm, the validity test rejects an interpretation candidate (a combination of trajectories) if two or more trajectories in the combination share measurements (see Section 5).

Algorithm 6 (EvaluateInterpretation)
Input: A set R of trajectories.

$T = null$;
for each interpretation candidate T' from R,
  perform validity test on T';
  if T' passes the test,
    construct interpretation BN model;
    compute $P(T'|D)$ using the BN;
    if $P(T' = correct|D)$ is the highest so far, $T = T'$;
return T;

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 [11] for a survey). We consider the complexity using the cluster tree method [30]. For $k = 6$, a good cluster tree has about 31 clusters. About one third of them each has a size of 7 variables. If variables acceleration (a), velocity (v), location (x, y) and measurement (x', y') each has a domain size of at least 10, then the belief state space of many clusters will be huge (about 10^7). Even if the inference computation is affordable, when it must be repeated for hundreds or thousands trajectories, it is very expensive and near real-time monitoring becomes impossible. Although a query DAG [12] 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 Figure 7 (a).

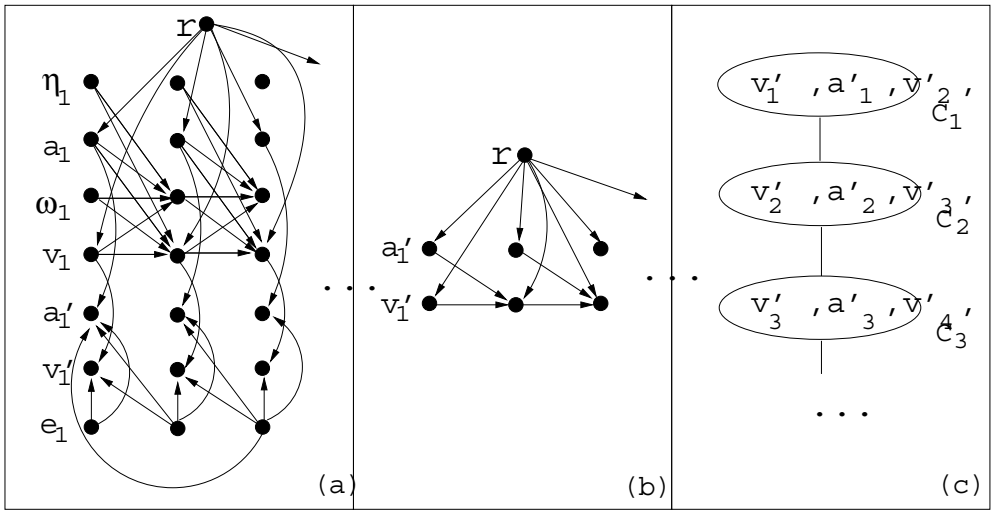

Figure 7: (a) Replacing location measurements in movement model. (b) Approximate movement model. (c) Cluster chain for movement model evaluation.

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 $\mathbf{r}, \omega_2, v_2$ and a_2 (using d-separation to (a)). However, if the value of

velocity and acceleration ($|v|$ and $|a|$) are larger enough than the value of measurement error ($|e|$), this dependence is not strong. By ignoring this dependence, we obtain the Markov property: a_1 and a_3 are independent given $\mathbf{r}, \omega_2, v_2$ and a_2, and v_1 and v_3 are independent given $\mathbf{r}, \omega_2, v_2$ and 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 structure in Figure 7 (b). In order to assign numerical parameters for the graphical model in (b) such that inaccuracy due to structural approximation can be minimized, we use the full model in (a) to compute *off line*

$$P(a_i'|\mathbf{r}) \quad and \quad P(v_i'|\mathbf{r}).$$

Note that if we had not made the independence approximation between ω, η and $\mathbf{r}$ in Section 4, we would not be able to obtain the model in (b), because ω and η would have to be represented as the child variables of $\mathbf{r}$, which would make the resultant model more expensive to use.

From (b), we obtain the cluster chain (c) (through the normal graphical operations as detailed in [30]), which can be evaluated efficiently as derived below: Conceptually, we follow the cluster tree method [30]. After initialization, the cluster C_1 is associated with the belief table (a probability distribution) $P(v_1', a_1', v_2', \mathbf{r})$. Other clusters have its belief 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 [30], 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. The benefit of multiple entering is explained below.

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

$$\frac{\sum_{v_1', a_1'} P(\mathbf{r}, v_1', a_1', v_2'|\nu_1, \alpha_1, \nu_2)}{P(\mathbf{r}, v_2')}.$$

This is a distribution over $\mathbf{r}$ and v_2'. At C_2, the local belief table is updated into the following 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, it is sufficient for the message from C_1 to C_2 to be

$$\frac{\sum_{v_1', a_1', v_2'} P(\mathbf{r}, v_1', a_1', v_2'|\nu_1, \alpha_1, \nu_2)}{\sum_{v_2'} P(\mathbf{r}, v_2'|\nu_2)},$$

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. Note also that to allow reasonably refined movement representation, v_2' normally has a large domain size (we used 12 in our experiments). Therefore, by replacing the message from a distribution over $\mathbf{r}$ and v_2' to a distribution over $\mathbf{r}$ only, the size of the message from C_1 to C_2 is reduced significantly. The amount of computation associated with the message passing is also reduced significantly as a result.

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

$$\sum_{v_2'} P(\mathbf{r}, v_2' | \nu_2) = d\ P(\mathbf{r} | \nu_2),$$

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 7 (EvaluateFullTrajectory)
Input: $\nu_1, ..., \nu_{k-1}, \alpha_1, ..., \alpha_{k-2}$ *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* $P(\mathbf{r} | \nu_1, ..., \nu_{k-1}, \alpha_1, ..., \alpha_{k-2})$;
return $P(\mathbf{r} | \nu_1, ..., \nu_{k-1}, \alpha_1, ..., \alpha_{k-2})$;

Using this algorithm, it is no longer necessary for the on-line inference computation to actually maintain the cluster chain. This contributes significantly to real time or near-real-time evaluation because a large number of evaluations must be performed. To obtain the parameters required by Algorithm 7, 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 Figure 7 (a). For our experiment (reported in Section 11), the off-line computation took about 12 hours using a SUN Ultra60. The large amount of computation is due to the large space in some distributions (for instance, $P(v_{i+1} | v_i, \omega_i, a_i, \eta_i, \mathbf{r})$ involves several variables each of which has a domain size of about 10).

Note that Algorithm 7 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 5, 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 Section 5). 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 Section 6 and assume no missing measurements at $1 < t < k$, a mistake will be made because the four measurements in the upper track are not qualified as a partial track and will be considered as noise. 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. Similar ideas have been explored in [33, 34] in auditory signal processing.

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.

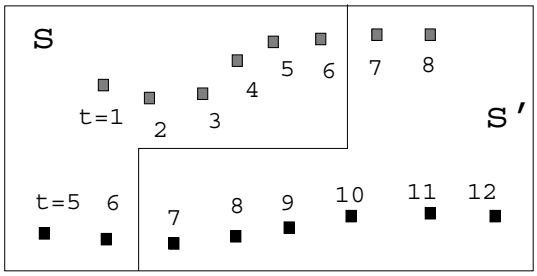

Figure 8: Two tracks across two scenes.

Consider the two tracks in Figure 8 which span two scenes: s with $t = 1, ..., 6$ (left) and s' with $t = 7, ..., 12$ (right). 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 Putting Pieces Together

Algorithm 8 connects the techniques that we presented so far together to interpret a given scene.

Algorithm 8 (InterpretScene2)
Input: A scene D.

```
1  run GetAllIsland(D) to decompose D into multiple islands;
2  for each island L, do
3      if no measurement for two or more instants in L,
4          interpret L as noisy measurements;

5      if no measurement for one instant in L,
6          run EvaluatePartialTrajectory() to evaluate each partial trajectory;
7          run InterpretIsland() to generate island interpretation for L;

8      if having measurements for each instant in L,
9          if L is a small island,
10             run EvaluateFullTrajectory() to evaluate each full trajectory;
11             run EvaluateInterpretation() to generate island interpretation T for L;
12             if P(T = correct|L) is not sufficiently high,
13                 run EvaluatePartialTrajectory() to evaluate each partial trajectory;
14                 run EvaluateInterpretation() to generate island interpretation T for L;
15         else /* L is a large island*/
16             generate full tracks based on peninsulas;
17             run EvaluateFullTrajectory() to evaluate each full trajectory;
18             run EvaluateInterpretation() to generate island interpretation T for L;
19             if P(T = correct|L) is not sufficiently high,
20                 run EvaluatePartialTrajectory() to evaluate each partial trajectory;
21                 run EvaluateInterpretation() to generate island interpretation T for L;
22 return collection of island interpretations as scene interpretation;
```

Line 1 decomposes the scene into islands using Algorithm 2. Subsequent processing is performed independently on each island. Three different cases are considered: missing measurements in two or more instants, missing measurements in one instant, and having measurements for each instant. Each case is handled by an *if* segment. Note that the three cases correspond qualitatively to: too many measurement-missing instants, very few, and none. The exact boundary (2 and 1) used in the algorithm is not significant.

Lines 3 and 4 deal with islands that miss measurements in too many time instants. These islands are likely to be produced due to noise and will be interpreted as such. Lines 5 through 7 handle islands that miss measurements in very few instants. We will come back to them later.

Lines 8 through 21 handle islands that have measurements in each instant. Note that missing measurements is still possible. It's just that for each time instant, at least one measurement has been obtained. Small and large islands are separately processed (by the *if* and *else*) segments. For small islands, initially each full trajectory is evaluated using the

reduced movement model (Section 8) and Algorithm 7. Algorithm 6 is used to evaluate each island interpretation constructed from only highly evaluated full trajectories. Note that trajectory evaluation results are reused during interpretation evaluation.

Lines 12 through 14 applies re-analysis when the interpretation obtained from only full trajectories is not sufficiently convincing. Each partial trajectory is now evaluated and a new set of interpretations containing partial trajectories are considered. Note that EvaluatePartialTrajectory() is a slight variation of Algorithm 7.

Lines 15 through 21 handle large islands. The processing is essentially the same as that for small islands, except that only trajectories consistent with peninsulas are evaluated. This restriction is not necessary for small islands but reduces computation significantly for large islands.

Next, we go back to lines 5 through 7. Since some instant has no measurement at all, all trajectories are partial. Note that this segment can be elaborated with the separate processing based on small or large islands, as described above. We did not do so to keep the presentation simple. Furthermore, if more than one instant have missing measurements, initial processing can be applied to trajectories with only one missing measurement. Re-analysis can be used if necessary to process additional trajectories with more missing measurements.

11 Experimental Results

To test the ideas and algorithms presented, we implemented a *scene simulator* and a *scene interpretor*. The simulator generates randomly a scene which is used as the input to the interpretor. The performance of the interpretor can then be evaluated by comparing its interpretation with the simulated tracks.

Each measurement simulated contains a 2-D location plus a signal frequency as would appear in passive sensing. The simulator allows specification of 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. This allows testing of the interpretor under different conditions.

In the experiment, we used a 200×200 grid region, populated by between 5 and 18 vehicles of two possible types. Note that the size of the grid and the number of vehicles are insignificant to the performance, but the density of vehicles is. A scene with 5 vehicles in the given grid is sparse and easy to interpret. A scene with 18 vehicles is dense and is much harder to process. The type 1 vehicle has its velocity from the set $\{6, 7, 8, 9, 10\}$ and the type 2 vehicle from $\{8, 9, 10, 11, 12\}$. The two sets have significant overlapping to increase the difficulty in vehicle type recognition. The initial velocity of each type is distributed evently. The accelerations of the two types of vehicles are chosen from $\{0, 1, 2\}$ and $\{0, 3, 4\}$, respectively. Measurement errors are chosen from $\{0, 1\}$ and can be added to the actual vehicle location in each of the 4 possible directions. Each measurement is associated with a signal frequency reading and an energy level. The frequency ranges from $\{100, 200, 300\}$ for type 1 and $\{300, 400, 500\}$ for type 2. The energy level ranges from $\{0, 2, 3, 4\}$ for type 1 and $\{0, 1, 2, 3\}$ for type 2. 'Negative' noise (missing measurement) is simulated when the energy level of a measurement is 0. 'Positive' noise is simulated by adding an arbitrary measurement in a time instant with 30% probability.

Figure 1 (top) shows a typical simulated scene in a 200×200 grid region of 20 vehicle tracks with two types of vehicles. The temporal length of the scene is $k = 6$. The interpretation generated (Figure 1 (bottom)) matched the simulated tracks with 100% accuracy.

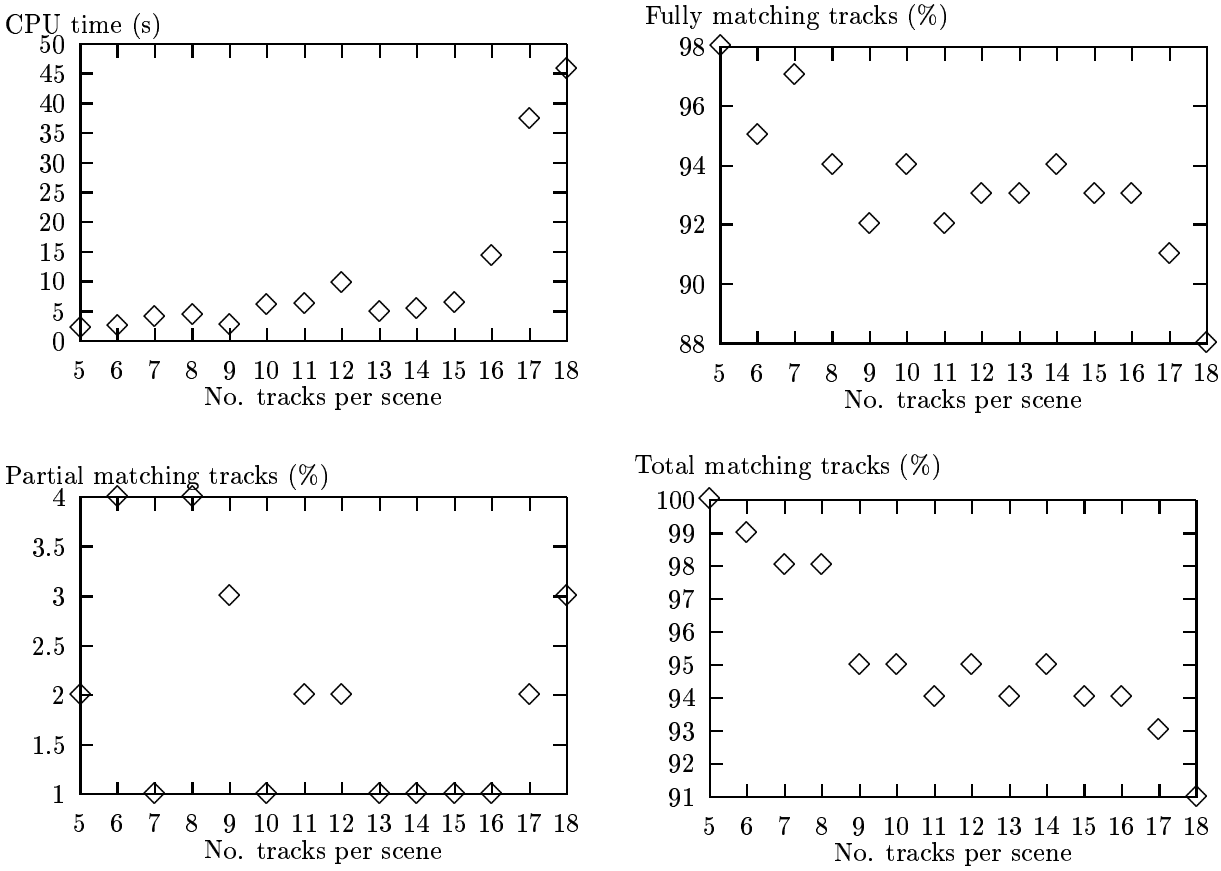

Figure 9: Summary of experimental results. X-axis: number of simulated tracks per scene in each batch. Top left: Average CPU time per scene. Top right: Percentage of fully matched tracks in each batch. Bottom left: Percentage of partially matched tracks in each batch. Bottom right: Percentage of both fully matched and partially matched tracks in each batch.

Figure 9 summarizes the interpretation results of 280 scenes of different degrees of difficulty. The 280 scenes were divided into 14 batches. Each batch has 20 scenes of the identical size (the number of simulated tracks). The x-axis of each graph in Figure 9 is labeled by the size (from 5 to 18) of the scenes in each batch. The larger the size, the higher the density of measurements and the more difficult to interpret the scene. Each vehicle 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).

The top left graph shows the average CPU time in interpreting each scene in a given batch. Our implementation is in Java using a Pentium II 400MHz PC. Near-real-time performance was obtained for a wide range of scene sizes. As the complexity of the scene increases, the interpretation time used increases gradually. Significant increases are observed after the scene size is larger than 16 as very larger islands are frequently detected in the scenes.

The top right graph shows the percentage of fully matching tracks in each batch. An interpreted track $\mathbf{r}'$ fully matches a simulated track $\mathbf{r}$ (which may have missing measurements) if $\mathbf{r}'$ matches each measurement in $\mathbf{r}$. The bottom left graph shows the percentage of

partially matching tracks in each batch. An interpreted track $\mathbf{r}'$ partially matches a simulated track $\mathbf{r}$ if $\mathbf{r}'$ matches each measurement in $\mathbf{r}$ except one. The bottom right graph shows the percentage of both fully and partially 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%.

12 Constructive Graphical Models Approach

So far we have focused on a constructive graphical models approach to the problem of vehicle monitoring. The problem, however, is an instance of a class of problems that some refers to as *sensor interpretation* problems [8]. In this class, a set of sensor data is generated by multiple events (e.g., multiple vehicles), but the number of events (e.g., the number of vehicles), the nature of each event (e.g., the type of each vehicle), and which pieces of data are produced by which particular event are unknown. The task is to identify the number and the nature of the events given the sensor data.

In the following, we generalize the constructive graphical models approach to sensor interpretation problems. We consider a domain where a unknown number of unknown events of a target class C occur. These events are assumed independent of each other.[2] The events generate a set D of observations. The task is to identify from D the events occurred which we refer to as the interpretation of D.

More formally, an *interpretation* T of D consists of a set of target events $\{e_1, e_2, ..., e_m\}$ in C and a partition $\{r_1, r_2, ..., r_m, N\}$ of D, where r_i is a subset of observations (believed to be) caused by the event e_i, and N is a subset of observations (believed to be) caused by events outside the target class C. Since the number m, the identity of each e_i, the composition of each r_i and with which event it associates are unknown, the number of possible interpretations resulting from all possible combinations is intractable.

The constructive graphical models approach starts with the Bayesian formulation: to compute the interpretation T^* such that the posterior probability $P(T^*|D)$ is maximal. Note that we overload T^* with two meanings as described in Section 2. To make the computation tractable, the approach deploys a number of techniques:

The general technique of *domain decomposition* separates the problem domain into independently or semi-independently evaluable subdomains each of which contains a subset of target events. It is elaborated by *island decomposition* and *peninsula decomposition* described below.

Given D, denote the set of events in T^* by $\{e_1^*, e_2^*, ..., e_m^*\}$ and the observation partition by $\{r_1^*, r_2^*, ..., r_m^*, N^*\}$. The technique of *island decomposition* partitions D into a set of islands $\{L_1, L_2, ..., L_k\}$ with the following properties:

1. Each island L_i is the union of one or more r_i^*'s plus some elements of N^*. That is, the island partition is consistent with the partition $\{r_1^*, r_2^*, ..., r_m^*, N^*\}$ but may be less refined.

2. The events $e_1^*, e_2^*, ..., e_m^*$ can be grouped into $E_1, E_2, ..., E_k$ accordingly such that observations in L_i depend only on events in E_i and events in E_i can only produce observations in L_i.

[2] Otherwise, the dependence can be explored and the complexity of the solution may be reduced.

It is important to indicate that the island partition $\{L_1, L_2, ..., L_k\}$ must be obtained without yet knowing T^*. Instead, the above properties of islands facilitate the computation of T^* from D. As exemplified in Eqn (1), island decomposition allows $P(T|D)$ to be evaluated as follows, where E_i is overloaded similarly as $T*$ (and T):

$$P(T|D) = P(E_1|L_1) \ldots P(E_k|L_k).$$

The amount of computation is reduced by excluding from explicit consideration those interpretations that relate observations on different islands to a same event.

The *peninsula decomposition* is applied to each island L to facilitate the computation of $P(E|L)$. Without lossing generality, denote the events in T^* that generate L by $\{e_1^*, e_2^*, ..., e_q^*\}$. Observations in L are divided into two sets: the initiator set $Init$ and the remainder set $Rest$, and observations in $Init$ are further partitioned into $\{Init_1, Init_2, ...\}$ with the following properties:

1. Each event e_k^* generates observations in exactly one initiator $Init_k$.
2. Each initiator $Init_k$ uniquely determines a subset $Rest_k \subset Rest$.
3. The union $A_k = Init_k \cup Rest_k$ is called a *peninsula*. A_k contains all observations that the event e_k^* generates.

Again, the initiators must be computed without yet knowing $\{e_1^*, e_2^*, ..., e_q^*\}$. The above properties of peninsulas facilitate the computation of these events. A peninsula A_k ensures that the event e_k^* can be identified by considering only observations in A_k (at this level of analysis). That is, it is either certain or highly probable that e_k^* maximizes $P(e|A_k)$ among other conceivable events e given A_k. Peninsula decomposition reduces the amount of computation by excluding from explicit consideration those interpretations that relate observations on different peninsulas to a same event.

The technique of *model separation* allows submodels to be constructed at different abstraction levels and their evaluation to be reused and combined. Given an interpretation, a graphical model can be constructed to represent the probabilistic dependence of the observations on hypothesized events as well as the overall coherence of the interpretation. To apply model separation, it is important that the dependence between an event and the observations that it generates is encoded in a graphical submodel that is sufficiently self-contained. The submodel that represents the overall coherence of the event set should have a concise interface with the submodel corresponding to each individual event. The individual event submodels should have no coupling other than the interaction through the coherence submodel.

With the graphical model so constructed, when an event participates in multiple interpretations, its contribution to the evaluation of $P(T|D)$ can be computed once and reused for each relevant interpretation. When many individual events participate in many interpretations, model separation saves the repeated computations for each interpretation.

The technique of *model approximation* reduces the amount of computation by simplifying an exact model. Often, a graphical model, reflecting all relevant dependence of the domain, may be too expensive to evaluate. The problem is worsened when a very large number of models (one per interpretation) must be evaluated. It is often the case that a simplified model is much more efficient to evaluate and can lead to the same decision as to accept or to reject a given interpretation even though the model evaluates $P(T|D)$ only approximately.

A graphical model can be simplified in a number of ways. Variables (nodes) that correspond to neither observations nor events may be eliminated. Dependence relations (links) that are weak may be ignored as if their endpoints (variables) are conditionally independent. A variable representing an observable quantity and another variable representing the corresponding observed value may be merged, which is equivalent to ignoring the observation errors. A continuous variable may be approximated by a discrete one. A discrete variable with a large space may be approximated by a variable with a small number of possible values.

The technique of *model compilation* identifies reusable intermediate computational results, computes them off-line, and uses them to reduce the on-line computation and to improve the performance. Model compilation may perform an off-line computation to simplify the model to be used for on-line computation. Hence, model approximation and model compilation may overlap. A key difference is that model compilation does not usually introduce error while model approximation usually does.

The technique of *re-analysis* allows more efficient computation at low abstraction levels and allows mistakes made to be corrected with limited and focused additional computation. The technique restricts the initial computation to that needed by a smaller subset of likely interpretations and only the corresponding graphical models or submodels will be evaluated. This restricted initial computation improves efficiency in most cases. As the results from the evaluation of low abstraction level submodels are being integrated at high abstraction level submodels, if there are indications that the initial subset of interpretations has been mistaken, the computation will be resumed at the low abstraction level with the target interpretation set enlarged. Such re-analysis corrects occasional mistakes made in restricted initial computation and improves the accuracy of the final interpretation.

13 Related Work

Vehicle monitoring has been a subject of considerable literature in the field of multiple target tracking and data fusion, mostly studied in the engineering discipline. Common approaches include Kalman filter, particle filter and multiple model method, and have various degree of success in a range of applications such as military target recognition, military situation assessment, smart weapons, machinery monitoring, automated plant management, and image processing. We make no claim that our approach is overall superior than the existing methods for vehicle monitoring: A systematic comparison is beyond the main objective of this work and a reasonable space limit. The primary contribution of this work is to show that, contrary to the previously held belief [8], graphical models, as a concise and intuitive representation and inference formalism, are applicable to sensor interpretation types of problems, where a single graphical model (or a few) is inadequate no matter it is constructed a priori or a posteriori. We have demonstrated this assertion by successfully applying a constructive graphical models approach to solving vehicle monitoring as an example problem. This contribution enables and encourages the application of graphical models to other problems of similar types, as well as to refinement and extension of our solution to vehicle monitoring.

For the sake of completeness, we briefly review some common approaches to vehicle monitoring. A study conducted in early 90s surveyed about 30 systems and identified more than 75 algorithms [25]. Our review focuses on the main ideas of some common approaches and is by no means exhaustive.

Most common approaches represent the domain by a set of state equations and a set of measurement equations. State equations model the vehicle movements and measurement

equations model the dependence of measurements on the movements. The task is to associate measurements with the vehicles (data association) and to estimate the vehicle tracks.

The nearest-neighbor standard filter [3] selects the measurement closest to a predicted measurement and uses it to estimate the corresponding vehicle location. Alternatively, the probabilistic data association filter [19] assigns an association probability for each close measurement and uses these probabilities to weight the measurements for location estimation. Kalman filtering [32] has long been applied to vehicle monitoring [3, 2]. The state equations and measurement equations are linear equations subject to zero-mean, white, Gaussian noise. At each time instant, vehicle states for the next instant are predicted using the state equations based on estimated states of the current instant. Measurements for the next instant are predicted using measurement equations and are compared with the actual measurements. The difference is then used to update the prediction and to arrive at the estimated states for the next instant. Kalman filtering is optimal when the linearity/Gaussian assumptions hold. To deal with nonlinearity due to vehicle manoeuvre, extended Kalman filters linearize the state equations by expanding the nonlinear functions in Taylor series and use the lower order terms [14].

Multiple model method attempts to deal with nonlinearity or 'mode jump' using multiple filters. For each time instant, multiple sets of representations are maintained each corresponding to one possible mode of the vehicle. The overall state estimate is a combination of the estimates from all filters. Static multiple model method [39] uses a fixed set of element filters without interaction. Interacting multiple model method [4] allows a fixed of element filters to interact. Variable structure multiple model method [38] uses a variable set of elment filters.

Performance of extended Kalman filters still rely on the Gaussian assumption. Particle filters [23] are more appropriate for nonlinear and non-Gaussian domains. The posterior probability distribution of the vehicle is represented by a set of random samples (the particles) with associated weights. As the number of particles becomes very large, this Monto Carlo representation approaches the optimal Bayesian estimate.

The constructive graphical models approach we presented provides an alternative to the above engineering approaches to address the nonlinear domains.

In the artificial intelligence field, vehicle monitoring has been chosen as a testbed for various methodologies and architectures for building knowledge-based systems. These efforts are mostly connected to the Distributed Vehicle Monitoring Testbed (DVMT) project [37, 15, 8, 45]. The approaches taken are based on incremental vehicle track construction combined with ad-hoc measures of uncertainty on hypothesized tracks. Heuristics and incremental planners are used to control the initiation and maintenance of tracks. Blackboard-based architectures are used to pursue multiple hypothesized tracks.

The idea similar to our re-analysis were explored in traditional knowledge-based systems [17, 16, 33, 34]. They applied approximate bottom-up processing to produce a set of likely high level models which were used to drive focused and more detailed processing in heuristic problem solving. The current work makes the idea applicable in a Bayesian, graphical models framework.

Since early 1980s, graphical models [43, 41, 36, 30, 9, 10, 46] have been actively researched as intuitive, concise, and normative formalisms for uncertain knowledge representation and inference in intelligent systems. Although building a graphical domain model a priori is the common way to apply this technology, recent advances on knowledge-based construction have allowed more flexible situation specific representations such as in story understanding or

military situation accessment [6, 21, 22, 28, 44, 24, 40, 31]. For the particular tasks addressed, it is usually sufficient to construct and evaluate one model after observations and the query variables are given. Our work has been influenced by their approach. However, the vehicle monitoring problem and the more general sensor interpretation problems pose additional representational and computational challenges by requiring construction and evaluation of a very large number of models in real or near-real time. Most of the techniques we presented are developed to meet such challenges.

Another branch of related work is dynamic Bayesian networks (DBNs) [13], where a dynamic system is represented by a sequence of BNs, each of which represents the dynamic domain at a given time instance. Forbes et al [18] used DBNs for modeling and control of automated highway vehicles. Boyen, Friedman and Koller [5] study how to discover DBNs from data. Pavlovic, Frey, and Huang [42] use a mixed-state DBN framework to model and classify object trajectories. Our work complements theirs on dealing with a large unknown number of moving objects.

14 Conclusion

We propose a constructive graphical models approach to solve a class of sensor interpretation problems and we demonstrate a solution to a particular problem instance, vehicle monitoring. To meet the computational challenges imposed by these tasks, we integrate a set of techniques: *Domain decomposition* separates the problem domain into independently or semi-independently evaluable subdomains each of which contains a subset of target events. *Model separation* allows submodels to be constructed at different abstraction levels and their evaluation to be reused and combined coherently. *Model approximation/compilation* helps to reduce the amount of computation to be performed within the allowable time frame. Domain decomposition, model separation and model approximation demonstrate different ways to explore independence inherent in the problem domain in order to speed up interpretation computation. *Re-analysis* allows more efficient computation at lower abstraction levels and allows mistakes made to be corrected with limited and focused additional computation. Variants of these techniques have been explored elsewhere under different contexts. We bring them together to solve the sensor interpretation problems in a Bayesian, graphical models framework.

The contribution of this work is manifold: Our result invalidates previously held belief in the field of knowledge-based systems which regarded graphical models not applicable to sensor interpretation problems. We have shown that by applying domain decomposition, model separation, model approximation/compilation, and re-analysis, these tasks can benefit from graphical models and real or near-real-time performance can be obtained. Our approach complements state equation based approaches in solving sensor interpretation types of problems by applying the graph modeling tools. Our approach extends previous work in situation specific graphical models to problems with a magnitude higher complexity.

Our approach as presented is centralized. As it is closely related to the traditional knowledge-based systems approach, which has provided many lessons in solving sensor interpretation problems through a distributed and multi-agent architecture, a natural step forward is to extend this approach under a multiagent framework.

Acknowledgements

This work is supported by the Research Grant OGP0155425 from the Natural Sciences and Engineering Research Council (NSERC) of Canada, by the National Science Foundation (NSF) under Grant No. IIS-9812755, and by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-99-2-0525.

References

[1] B. Abramson. Arco1: an application of belief networks to the oil market. In B. Dambrosio, P. Smets, and P. Bonissone, editors, *Proc. 7th Conf. on Uncertainty in Artificial Intelligence*, pages 1–8, Los Angeles, CA, 1991.

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

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

[4] H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Trans. Automatic Control*, AC-33(8):780–783, 1988.

[5] X. Boyen, N. Friedman, and D. Koller. Discovering the hidden structure of complex dynamic systems. In *Proc. 15th Conf. on Uncertainty in Artificial Intelligence*, pages 91–100, Stockholm, 1999.

[6] J.S. Breese. Construction of belief and decision networks. *Computational Intelligence*, 8(4):624–647, 1992.

[7] L. Burnell and E. Horvitz. Structure and chance: melding logic and probability for software debugging. *Communications of the ACM*, 38(3):31–41, 1995.

[8] 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.

[9] E. Castillo, J. Gutierrez, and A. Hadi. *Expert Systems and Probabilistic Network Models.* Springer, 1997.

[10] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, 1999.

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

[12] 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.

[13] T.L. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, (5):142–150, 1989.

[14] T.L. Dean and M.P. Wellman. *Planning and Control.* Morgan Kaufmann, 1991.

[15] E.H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic, Boston, MA, 1988.

[16] E.H. Durfee and V.R. Lesser. Incremental planning to control a time-constrained, blackboard-based problem solver. *IEEE Trans. on Aerospace and Electronic Systems*, 24(5):647–662, 1988.

[17] L.D. Erman, F A. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, 1980.

[18] 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.

[19] T.E. Fortmann, Y. Bar-Chalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Oceanic Eng.*, OE-8:173–184, 1983.

[20] A.S. Gertner, C. Conati, and K. Vanlehn. Procedural help in ANDES: generating hints using a bayesian network student model. In *Proc. of AAAI*, pages 106–111, Menlo Park, 1998.

[21] R.P. Goldman and J.S. Breese. Integrating model construction and evaluation. In D. Dubois, M.P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Proc. 8th Conf. on Uncertainty in Artificial Intelligence*, pages 104–111, Stanford University, 1992. Morgan Kaufmann.

[22] R.P. Goldman and E. Charniak. A language for construction of belief networks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(3):196–208, 1993.

[23] N. Gordon, D. Salmond, and A. Smith. Novel approach to non-linear/non-Gaussian Bayesian state estimation. 140(2):107–113, 1993.

[24] P. Haddawy. Generating bayesian networks from probability logic knowledge bases. In *Proc. 10th Conf. on Uncertainty in Artificial Intelligence*, pages 262–269, San Francisco, CA, 1994.

[25] D.L. Hall. *Mathematical Techniques in Multi-sensor Data Fusion.* Artech House, 1992.

[26] D. Heckerman. A tractable inference algorithm for diagnosing multiple diseases. In M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 163–171. Elsevier Science Publishers, 1990.

[27] D. Heckerman, J.S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, pages 49–57, 1995.

[28] M.C. Horsch and D. Poole. A dynamic approach to probabilistic inference using Bayesian networks. In *Proc. 16th Conf. on Uncertainty in Artificial Intelligence*, pages 27–29, Cambridge, MA, 1990.

[29] T. Huang and S. Russell. Object identification: a Bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103:77–93, 1998.

[30] F.V. Jensen. *An Introduction To Bayesian Networks.* UCL Press, 1996.

[31] E. Santos Jr. and E.S. Santos. A framework for building knowledge-bases under uncertainty. *Journal of Experimental and Theoretical Artificial Intelligence*, (11):265–286, 1999.

[32] R.E. Kalman. A new approach to linear filtering and prediction problems. *J. Basic Engineering*, pages 35–46, 1960.

[33] F. Klassner, V.R. Lesser, and H. Nawab. Combining approximate front end signal processing with selective reprocessing in auditory perception. In *Proc. of AAAI*, pages 661–666, Providence, Rhode Island, 1997.

[34] F. Klassner, V.R. Lesser, and H. Nawab. The role of data reprocessing in complex acoustic environments. In *Proc. of AAAI*, pages 997–1003, 1998.

[35] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In D. Geiger and P.P. Shenoy, editors, *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, pages 302–313, Providence, Rhode Island, 1997.

[36] S.L. Lauritzen. *Graphical Models.* Oxford, 1996.

[37] 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.

[38] X.R. Li and Y. Bar-Shalom. Multiple-model estimation with variable structure. *IEEE Trans. Automatic Control*, AC-41(4):478–493, 1996.

[39] D.T. Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Trans. Automatic Control*, pages 434–439, 1965.

[40] S.M. Mahoney and K.B. Laskey. Constructing situation specific belief networks. In *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, pages 370–378, 1998.

[41] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley and Sons, 1990.

[42] V. Pavlovic, B. J. Frey, and T. S. Huang. Time-series classification using mixed-state dynamic bayesian networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 609–617, 1999.

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

[44] D. Poole. Probabilistic horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[45] R. Whitehair. *A framework for the analysis of sophisticated control.* PhD thesis, University of Massachusetts, 1996.

[46] Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.

[47] Y. Xiang and H. Geng. Distributed monitoring and diagnosis with multiply sectioned Bayesian networks. In *Proc. AAAI Spring symposium on AI in Equipment Service, Maintenance and Support*, pages 18–25, Stanford, 1999.

[48] Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned Bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.