Sensing, Tracking, and Reasoning with Relations

Leonidas J. Guibas
Computer Science Department
Stanford University
Stanford, CA 94305

1 Introduction and Background

Suppose we have a set of sensor nodes spread over a geographical area. Assume that these nodes are able to perform processing as well as sensing and are additionally capable of communicating with each other by means of a wireless network. The sensors may include small video or still cameras, acoustic microphone arrays, PIR (passive infrared intrusion), seismic, or magnetic sensing devices. Examples of such integrated devices include the UCLA/Sensoria WINS nodes [34] or the Berkeley motes [28]. The sensors may be controllable in limited ways: they can activated, possibly aimed, commanded to sense and transmit data to their neighbors, etc. Some may be even mounted on mobile ground or air platforms and directed to move to specific locations. Though each node is an independent hardware device, they need to coordinate their sensing, computation, and communication in order to acquire relevant information about their environment so as to accomplish some high-level task. The integration of processing makes such nodes more autonomous and the entire system, which we call a *sensor net*, becomes a novel type of sensing, processing, and communication engine.

Such sensor nets can be deployed in a variety of civilian and military contexts, including factory automation, environmental monitoring, health-care delivery, security and surveillance, and battlefield awareness. Their advantages over alternate architectures have been argued, for example in [34]. Briefly, these include:

- deployment can be rapid, as no cabling infrastructure is needed,
- sensor placement close to signal sources results in better signal-to-noise (SNR) ratio,
- on-board processing allows the sensors to communicate compressed signal data,
- the system can be made robust against the failure of a small number of its sensor nodes, and
- a distributed implementation allows the system to scale gracefully — unlike sensor systems with centralized processing.

Unique to such sensor nets is the ability to cover wide areas that no single sensor could possibly observe and to provide dense spatial sampling with multiple aspect and sensing modalities. Most of all, the integration of processing with sensing allows dimensionality reduction on the acquired signals and the communication of information in a form that is far more compressed and symbolic.

Nevertheless, the distribution of sensing and processing comes with certain limitations on the processing and communication capabilities of such device. Power consumption will clearly be an issue, as such integrated sensors will typically be untethered or mounted on mobile platforms. There will also be communication range, bandwidth, and susceptibility to noise limitations due to the wireless technologies (such as radio links) commonly used with such devices. These limitations make it important that the software controlling the sensor net be built so as to optimally use the net's resources towards the task at hand. The goal of this paper is to present a methodology for planning and controlling the sensing, processing, and communication actions needed to accomplish a certain mission, while respecting the system resource constraints described above.

To be concrete, consider two settings. SET1 is people moving around a building; we are interested in security and surveillance applications, such as the detection of unusual activities and the monitoring of particular suspicious individuals. SET2 is a military engagement in an open terrain with a few buildings and ground enemy (e) and friendly (f) vehicles moving in it; a commander must make key decisions that depend on the world state. In both settings, the sensor net needs to provide useful high-level information to its clients.

In SET1 we may be interested in answering spatio-temporal queries such as:

- *are there groups of people moving coherently?*

- *track and report the location of the leader of group g for the next t minutes.*

In SET2 we may have queries of the following type:

- *is friendly vehicle f_{13} surrounded by enemy vehicles?*
- *which enemy vehicle is likely to be able to get close to building b_2 first?*

Such queries may also be a tool for organizing the sensor net itself. Indeed, suppose the sensors are actually mobile and each can talk to all its neighbors with a range. We may want to:

- *organize the sensors into clusters and maintain these clusters as the sensors move around.*

Such a clustering can be used to define a hierarchical structure on the nodes that conforms and adapts to the nodes' changing topology.

In this paper we propose a mathematical framework on how high-level queries, such as those above, can be transformed into low-level sensing, computation, and communication operations designed to produce the desired answers — while minimizing the power and other resources expended in satisfying the queries. As we illustrate below, this is an algorithm design problem with many new twists; additional work is needed to understand and elaborate all its ramifications. Note that several of the queries above refer to global, aggregate, and relational aspects of the environment. Indeed, though most sensing acquires data in the continuous domain, the information most useful to the system's clients is often of an aggregated or discrete nature. Thus by its nature the sensor net is a hierarchical hybrid system where sampled continuous signals transition to discrete symbolic information as we go up the task hierarchy. We argue that the interface between continuous and discrete data can be pushed to a very low level in the system architecture, with significant benefits in economy and speed.

Our focus will be on tracking spatial or temporal relations between objects and local or global attributes of the environment, as opposed to the detailed estimation of positions and poses of individual objects. High-level behaviors of objects may be trackable more robustly than their exact positions, relations between objects may be easier to track than each object separately, and the large-scale behavior of an ensemble of objects may be easier to ascertain than the motion of the individual objects. By focusing on relations and the logical structure of the evidence with respect to the task at hand, we will be able to allocate sensor and computation resources where they are most needed.

The paper is organized as follows. Section 2 discusses the possibility of sensing relations between objects at a low level and aggregating such information towards a useful end. Section 3 discusses the kind of algorithmic theory necessary to design and analyze such sensor algorithms. In Section 4 we present some considerations relevant to directly sensing non-local relations. Section 5 takes on the all important topic of updating information under object motion and shows how to do relational tracking. Section 6 discusses issues related to uncertainty and probabilistic reasoning and, finally, Section 7 concludes the paper.

2 Objects and their Relations

The purpose of a sensor system is often viewed as obtaining information that is as extensive and detailed as possible about the unknown parts of the world state. Any targets present in the sensor field need to be identified and localized. Their motion must be tracked and their communications intercepted. All this data is to be centrally aggregated and analyzed. This is a reasonable view when the potential use of this information is not known in advance, and when the cost of the resources needed to acquire and transmit the information is either fixed, or of no concern. Such a scheme, however, runs the danger of flooding the network with useless data and depleting scarce resources such as battery power and human attention, when that is a concern. There are obvious ways to be more selective in choosing what sensor nodes to activate and what information to communicate; protocols such as *directed diffusion* address exactly this issue [25].

But there are a number of other, less obvious ways that the sensor net may be effective yet economical at the same time:

- If the goal of identifying and localizing a few targets close to a sensor is to ultimately establish a spatial relation between them, it may less expensive for the sensor to *sense the relation directly*, rather than attempt full target localization.
- Partial information about the environment can be aggregated and saved in certain nodes. If this is done in a clever way, a variety of global queries can be answered just be combining these prestored answers, without requiring any additional sensing and without imposing a significant storage load on the nodes.

Let us look at some examples to see what savings might be possible.

LEADER_IN_THE_CORRIDOR: Consider the simple 1-D scenario in Figure 1, which takes place inside a building where two corridors meet near an exit. Individuals a, b, c, d, and e are running towards the exit, while trying to avoid individual m. They are being tracked by cameras 1, 2, and

3, as shown. The tracking system is interested in knowing who among these five individuals is leading the group, i.e., is ahead of everyone else and therefore most likely to exit first. The building and camera geometry is assumed to be known in advance. Although the sensor system could try to localize the individuals involved, the cameras can also directly sense spatial relationships between these individuals. In the configuration shown, camera 2 can sense that $a \succ b$, while camera 1 can sense that $b \succ c$, $c \succ d$, and $d \succ e$, where '$\succ$' indicates the 'AHEAD_OF' relation. These relations allow the sensor system to logically conclude that a is the current leader of the group.

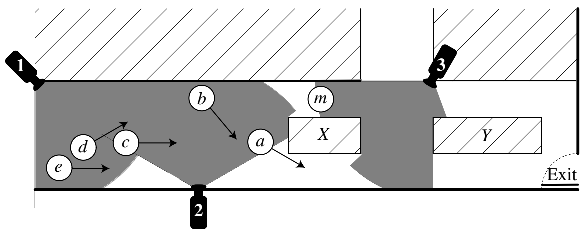


Figure 1: Tracking the leader in the corridor.

Note that the cameras may be able to ascertain these elementary spatial relationships with simple image processing, based just on visual features allowing identification of the individuals and without either full camera calibration or target localization. Assuming that the camera positions are known relative to each other, hand-off of individuals from one camera to another can also help establish such spatial relationships (say an individual enters from the left side of the viewing frustum and is therefore known to be 'to the left' of other individuals in the scene).

AM_I_SURROUNDED?: In this 2-D scenario, we have friendly and enemy vehicles present in a relatively flat terrain. Suppose that we want to know if friendly vehicle f is surrounded by enemy vehicles e_1, e_2, and e_3. For the sake of this exercise, let us use the following geometric definition of being surrounded: f is surrounded if there is no line in the plane that can separate f from all of e_1, e_2, and e_3. Such a configuration is shown in Figure 2. Suppose there exist three PIR sensors s_1, s_2, and s_3, located respectively inside the triangles fe_2e_3, fe_3e_1, and fe_1e_2. These sensors (perhaps using also some additional information available to the sensor net) sense that the vehicles are inside the cones shown in Figure 2. In each case not all of the cones emanating from the same sensor can be contained inside a single half-space supported by the sensor; this implies that the sensors can assert the geometric relations CCW(fe_2e_3), CCW(fe_3e_1), and CCW(fe_1e_2), where CCW(abc) is the relation that points a, b, and c in the plane form a *counter-clockwise* oriented triangle. From these three mathematical relations we can conclude that we also have CCW($e_1e_2e_3$) and that in fact friendly vehicle f is surrounded, in the sense given above.

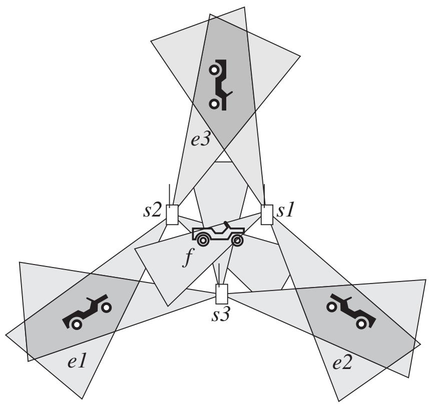
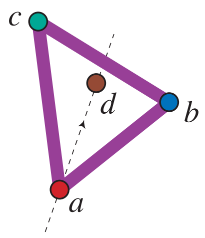


Figure 2: Determining if a vehicle is surrounded.

Note that in this example the enemy vehicles need not be precisely localized — a scaled version of this configuration can make the regions in which the enemy vehicles have been determined to lie arbitrarily large. Furthermore, the sensor net need only distinguish the friendly vehicle from the enemy vehicles, but not the latter among themselves. Finally observe that the well-placed PIR sensors that can resolve the AM_I_SURROUNDED query need not not be physically close to any of the vehicles involved.

The CCW spatial relations used above can be useful for answering a variety of other geometric queries. For instance, we may be interested in knowing what are the extremal members in a group of targets, such as the northernmost or westernmost targets. Overall all directions, the set of extremal targets corresponds to the *convex hull* of the target group, which is the smallest convex region guaranteed to contain all the targets. In Figure 3, the four CCW relations shown allow us to conclude that the convex hull of these four targets is abc. This is an artificially small example and in general, when many targets are present, only a small fraction of the targets will be on the convex hull. The convex hull can always be determined by an appropriate set of CCW relations of size linear in the number of targets [14]. In fact, the 'surrounded' condition above is equivalent to the friendly vehicle being in the convex hull of the enemy vehicles.

TARGET_COUNTING: Suppose we have n sensors fairly evenly spread over a 2-D area. Suppose further that these sensors are able to detect, localize, and count the number of targets in a small region around themselves, either in isolation or in a light-weight collaboration with their neighbors. This local region diameter is assumed to be comparable to the sensor spacing. Our global goal is to count the total number of targets present in certain types of simple geometric areas — and for the sake of this example assume that these areas must be half-spaces. The target count can easily



Proof of correctness:

- CCW(a, b, c)
- CCW(d, b, c)
- CCW(d, c, a)
- CCW(d, a, b)

Figure 3: Determining the convex hull of the targets.

be obtained in $O(n)$ time by interrogating and aggregating data from each of the sensors.

However, if we are willing to precompute and store some partial results, then we can do significantly better. Suppose we put a $\sqrt{n} \times \sqrt{n}$ grid over the sensor net so that in each of its cells there is at least one sensor whose sensing region contains the cell, as in Figure 4. We now activate the sensors and count the number of targets within each of the grid cells; furthermore, we propagate these counts along rows of the grid, so that in the end in each cell we know the total target count not only in the cell itself, but also the aggregate count for the cells to its left in its row, and to its right in its row.

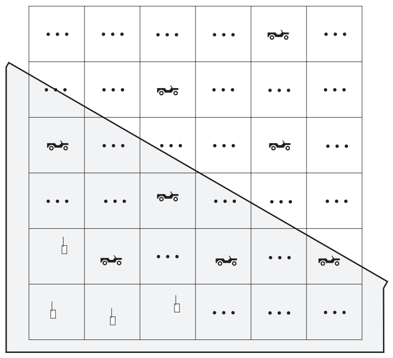


Figure 4: Counting the targets in a half-space.

Consider now a half-space query, as shown in Figure 4. Note that, except for the at most $2\sqrt{n}$ grid cells intersected by the half-space edge, all other cells are either fully contained in the range, or are fully outside the range. We can easily compute the total number of targets in the half-space by aggregating target counts from the partially covered cells, plus the relevant pre-stored totals for each row from their left or right neighbors, as appropriate. Thus, by using storage only for local targets and two counts in each node, we are able to answer the half-space target counting query in $O(\sqrt{n})$ time. Note that this applies to *any* half-space query: our preprocessing has exploited knowledge of the general shape of the query (half-space) but *not* of its exact parameters. By using more sophisticated geometric methods, the same can be done for target counting in areas that have other simple geometric shapes; furthermore the assumption about even distribution of the sensors can be dropped — as long as the union of the sensed regions cover the domain of interest. This falls under an area studied in computational geometry known as *range searching* [31, 3]. To our knowledge, little has been done on this in this direction in the sensor setting; the closest is work motivated by the data-base view [10, 11].

CLUSTER_MAINTENANCE: In this scenario we have n friendly vehicles carrying sensors. These vehicles wish to organize themselves into clusters, based on their current locations and a certain desired cluster radius (perhaps corresponding to a radio link range); see Figure 5. The cluster organization will allow for simpler communication protocols among vehicles in the same cluster, and avoid duplicate sensor measurements by nearby vehicles. In [17] we gave a distributed randomized algorithm that performs this type of cluster formation and yields a number of clusters that is a constant-fraction approximation of the optimum possible, with high probability. The algorithm is based on a 'cluster-head–nomination' protocol. We assume that initially each vehicle has a unique ID (UID) — these can be assigned by standard protocols (e.g. use the serial number of the vehicle), or by a random process. Each vehicle nominates as a cluster head the vehicle of highest UID that is within its radius (this might be itself). All nominated vehicles become cluster-heads and form a cluster with all their nominators (except for those that they themselves became cluster heads). This extremely simple protocol does not quite work, but as shown in [17], a very simple hierarchical variant of the same algorithm does. In the hierarchical version we use repeatedly the cluster nomination procedure with geometrically increasing radii, up to the final desired radius. At each round only the leaders already elected in the previous round participate.

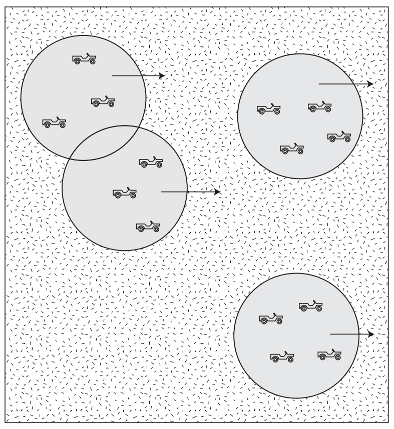


Figure 5: Clustering vehicles.

An interesting aspect of this algorithm is that it can be implemented without knowledge of the actual geographic positions of the vehicles. The protocol only needs for each node to be able to know what other nodes are within certain distance ranges from it. This could be done by broadcasting at different strengths and asking 'who is there?'. While not so important for vehicles that typically will have GPS devices, this observation means that the same algorithm, implemented using directly sensed distance relationships, could be made to run on much lower power mobile sensor devices.

Since motion and vehicles are inherent in all the above examples, the reader may wish to ask what happens when we wish to answer such queries not just once, but during a certain period in which motion occurs. This will be the topic of Section 5, where we will see how relational attributes can be updated incrementally.

3 Reasoning with Relations

If we knew the relevant manifest variables defining the world state (say the position and identity of each target), then computing the answers to queries such as those discussed above is a standard algorithm design problem. An algorithm typically proceeds by doing both numerical and relational (e.g., test) manipulations on these data, in order to compute the desired answer. The quality of the algorithm is judged via certain performance measures on resources used (time, space, etc.).

This classical algorithm/complexity view needs to be modified in the sensor net context. Many differences exist:

- the values of the relevant manifest variables are not known — they have to be sensed;
- elementary relations between such variables may be easier to determine than the values of the variables themselves;
- the cost of sensing different variables or relations of the same type can be vastly different — depending on the relative locations of targets and sensors, the sensing modalities available, and the communication costs;
- frequently the value of a variable, or a relationship between variables, may be impossible to determine using the resources available in the sensor net; however, alternate variable values or relations may serve our purposes equally well.

Thus we need a new mathematical theory of algorithm design that includes the cost of accessing the manifest variables of the problem, or of determining useful atomic relationships among them. Furthermore, these costs cannot be precisely known in advance and may only be estimated. In addition, there may be values and relations that have been independently determined by the sensor net (say, while processing other tasks, or during a preprocessing step) and can be made available to the algorithm at relatively low cost (communication), as in our target counting example. Thus the model needs to include both 'push' and 'pull' types of information flow.

To design an overall strategy, several key questions need to be addressed.

- what are the important objects in the environment to be sensed?
- what parameters of these objects are most relevant?
- what relations among these objects are critical to whatever high-level information we need to know?
- which is the best sensor to acquire a particular parameter?
- can relations between the objects be sensed directly?
- how many sensing and communication operations will be needed to accomplish the task?
- how coordinated do the world-models of the different vehicles need to be?
- at what level what do we communicate information, in the spectrum from signal to symbol?

Addressing these questions presents several challenges. Indeed, while the computational and communication complexity of different algorithms for the same problem can be assessed with standard techniques, the on-line nature of sensing will require the use of methods such as *competitive analysis* [36], or *the value of information* [24], or other sensor utility measures [35, 12] to account for the fact that the value of sensor readings cannot be known before they are made.

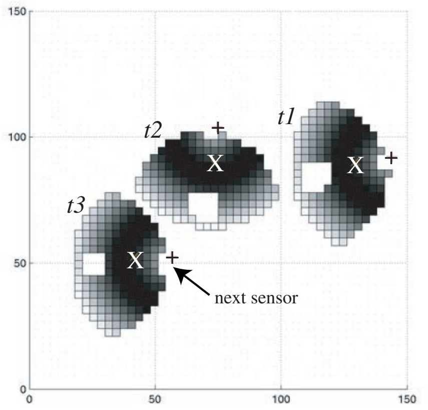
4 Sensing Non-Local Relations

In our LEADER_IN_THE_CORRIDOR example, the elementary $\succ$ ('AHEAD_OF') relations relevant to the problem are all between pairs of objects both visible from the same camera. Such 'local' relations are often all that we need to reason about a situation. However, local relations are not always sufficient. In the AM_I_SURROUNDED scenario, for example, each of the three CCW relations used are indeed sensed by a single PIR sensor. In general though, the sensor net may be tasked to try to ascertain if a CCW relation holds between three targets, without knowing that there is a PIR sensor that can see all three in the fashion shown — indeed, such a sensor may not exist. Yet by sufficiently localizing the targets using combinations of other sensors, the truth of the CCW predicate may still be determined with high confidence.

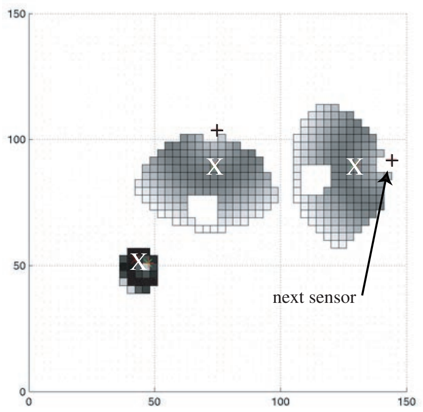
Suppose the sensor net has three probability distributions describing its belief about the locations of the three targets

t_1, t_2, and t_3. Using this information we can ask: 'what is the probability that CCW(t_1, t_2, t_3) is true?' Unless this probability comes out to be very close to 1 or to 0, the sensor net does not have adequate information about the targets to determine the validity of CCW(t_1, t_2, t_3) with high confidence. Note that the CCW relation may be uncertain because of extrinsic reasons, say by reason of the poor localization of the targets, or because of intrinsic reasons, when in fact the three targets are almost collinear. The sensor net may choose to expend additional sensing and communication resources so as to localize some of the three targets better, thus hopefully increasing the confidence that CCW is true or false. Or the sensor net may simply report that the (truth or falsity of the) relation cannot be determined with sufficient confidence, and thus the overall task manager should seek to establish the eventual goal by alternate means. The decision between the two options is delicate: the sensor net has to weight the potential benefit of the new information against the resources needed to acquire it.

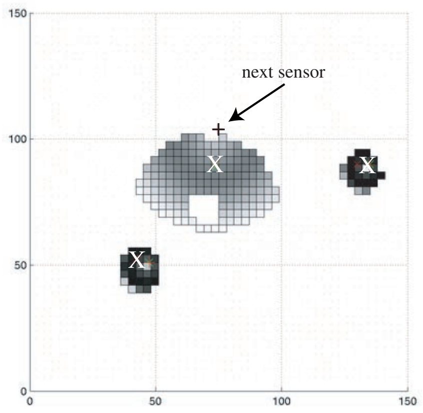
The sequence of four images below (Figure 6) shows a simulation of the belief state of the system on the locations of the three targets after successive sensor readings. A fairly dense and uniformly distributed sensor field of acoustic sensors has been assumed. For simplicity we use only amplitude sensors and therefore each sensor reading helps localize a target to within an annulus centered at the sensor. In the initial state two readings have been taken on each target. In each subsequent frame a sensor was chosen to reduce the location uncertainty of the most uncertain target. In each case the 'X' indicates the target location, the shaded boxes the system belief state distribution for a target, and the highlighted '+' the location of the next sensor to be used.



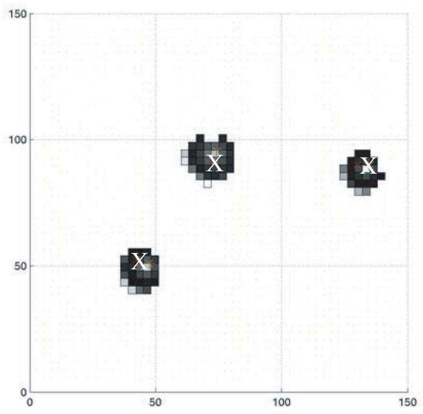
(2)



(3)



(4)



(5)

Figure 6: Resolving the CCW relation.

By frame (5) the targets have been localized sufficiently for resolving the CCW relation. This is shown in Figure 7 that displays the amount of probability mass in the product space shifting from the uncertain bin (gray) to the counterclockwise bin (white) with each additional sensor reading.

A further discussion of the fundamental issues in incorporating probabilistic reasoning in our sensing and tracking algorithms is given in Section 6.

It is interesting to investigate algorithms for optimal sensor selection in order to reduce uncertainty in the relations we are trying to establish. In our CCW example above, this can

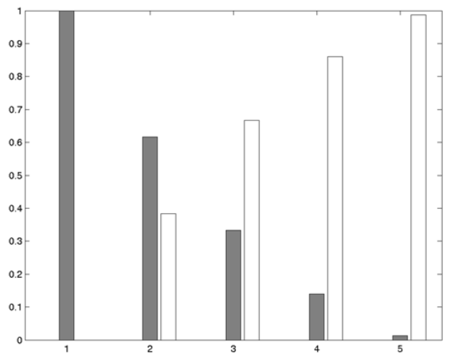


Figure 7: Reduced CCW uncertainty after each sensor reading.

arise because the distributions for t_1, t_2, and t_3 allow a sufficient mass of triplets of possible target locations, one each for t_1, t_2, and t_3, that are collinear or possibly clockwise oriented. Consider now the situation in Figure 8. Note that PIR sensor s_1 may look at one of the tails of the distribution for target t_1 and, upon seeing nothing there, lop off a large chunk of this distribution and reduce its spread. Yet that reduction is almost useless as far as eliminating wrongly oriented triplets of possible target locations. Another PIR sensor s_2 may lop off a smaller part of the t_2 distribution, yet have a much more significant benefit towards certifying CCW(t_1, t_2, t_3). This example shows that the choice of which sensors to activate in order to obtain quality information about an uncertain relation is subtle. Furthermore, the sensor net has to take into account both the expected information gain from the sensor reading, as well as the cost of conveying the information from that sensor to the location where the CCW predicate is to be certified. Such trade-offs between information quality and communication costs arise even when only a single target needs to be localized and have been studied in the *information-driven sensor querying* (IDSQ) framework of [13].

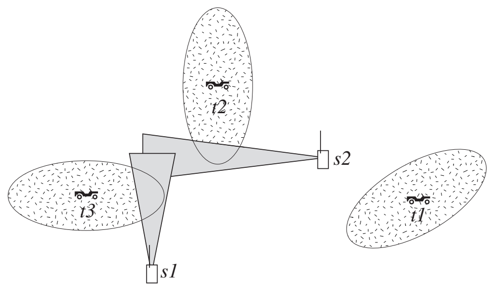


Figure 8: The effect on CCW of localizing targets.

5 Incrementally Updating Information

In many contexts we may be interested not just in answering a question about the state of the world once, but in continuously monitoring the answer to the query over a period of time. Let us refer to the quantity we seek to track over time as the *attribute of interest*. In such situations it behooves us to try to maintain the value of this attribute incrementally, rather than recomputing it from scratch at each step. One advantage of the relation-based attribute computations we propose is that the defining objects can move, but as long as the relations of interest among them stay valid, our attribute computation also remains valid. Of course, at some point, one of the supporting relations between objects may either become invalid, or it may still be valid but the information available to the sensor net may be insufficient to certify its validity. When such events happen, we can hope that either a different support set for the value of the attribute can be easily established, or that the value of the attribute itself can be updated incrementally.

Consider again out LEADER_IN_THE_CORRIDOR example. Suppose that individual d moves ahead to overtake c, as in Figure 9. Camera 1 can no longer support the relations $b \succ c$ and $c \succ d$ — however, camera 1 can support the relation $b \succ d$ and camera 2 can support the relation $b \succ c$. It follows that the conclusion that a is the leader is still valid: all the other nodes can be reached from a by following chains of '$\succ$' relations.

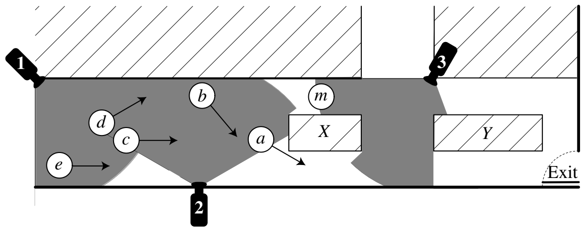


Figure 9: Maintaining the leader in the corridor.

In general we can think of such situations as follows. A computation (or frequently just the value) of the attribute we are interested in is *certified* by a number of elementary relations among the objects involved. We call these elementary relations *certificates* and think of them as a cache of assertions about the world whose job is to simplify the computation of our attribute. As objects move and relations fail (or become unsupportable by the sensors), we must update our assertion cache so that at all times we have a valid computation of the attribute of interest. Exactly how the cache should be updated by seeking new relations supportable by the sensors is problem dependent. This assertion cache design problem is addressed by the framework of *Kinetic Data Structures* [22, 8] (or KDS for short), which are data structures aimed for problems dealing with objects in motion. When the sensors can no longer support one or more of the certificates in the assertion cache, a KDS algorithm is invoked to try find an alternate certification of the attribute computation, or to repair the computation incrementally and certify the new computation as well; see

Figure 10.

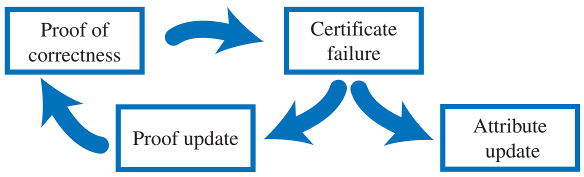


Figure 10: The inner loop of a kinetic data structure.

Look again at the simple convex hull example we showed in Section 2. Suppose that targets a, b, and c are stationary and only target d is moving as shown in Figure 11. At some point the certificate CCW(d, b, c) will no longer be supported by the sensors and soon afterwards its negation, CCW(c, b, d) will be supported (the CCW predicate flips its value whenever two of its arguments are transposed). In this case the KDS repair mechanism will discover that the convex hull has actually changed and is now $abdc$, a conclusion supported by the certificates shown.

Old proof	New proof
CCW(a, b, c)	CCW(a, b, c)
$\boxed{\text{CCW}(d, b, c)}$	$\boxed{\text{CCW}(c, b, d)}$
CCW(d, c, a)	CCW(d, c, a)
CCW(d, a, b)	CCW(d, a, b)

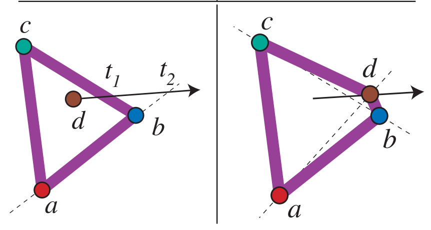


Figure 11: Updating the convex hull of the targets.

If the certificate set is chosen judiciously, the KDS repair can be a local, incremental process — a small number of certificates may leave the cache, a small number may be added, and the new attribute certification will be closely related to the old one. A good KDS exploits the continuity or coherence of motion and change in the world to maintain certifications that themselves change only incrementally and locally as assertions in the cache fail. The design of a good certificate set to maintain presents an interesting trade-off. On the one hand we must keep enough information about the state of the world to allow a quick, incremental recomputation of the attribute. One the other hand, too much information can be costly, as it will force the algorithm to process many certificate failures that are possibly irrelevant to the attribute of interest. Thus, in the same way that classical data structures need to balance the efficiency of access to the data with the ease of its update, kinetic data structures must tread a delicate path between 'knowing too little' and knowing too much' about the world. A good KDS will select a certificate set that is at once economical and stable, but also allows a quick repair of itself and the attribute computation, when one of its certificates fails.

A KDS may also exploit knowledge about the motions of the objects involved, their so-called *motion plans*, to avoid continuously checking some certificate conditions. Information about the motions, either known *a priori* or estimated from the observed data, can be used to generate conservative approximations as to when certificates might fail, and therefore eliminate needless continuous sensing and the power consumption associated with that. Thus, typically associated with a KDS is an *event-queue mechanism* in which certificates are placed that cannot fail until sometime in the future.

A number of measures have been proposed by which to evaluate the quality of a KDS for a specific problem. These include:

responsiveness: a KDS is called *responsive* if the cost, when a certificate fails, of repairing the certificate set and updating the attribute computation is small;

efficiency: a KDS is called *efficient* if the number of certificate failures it needs to process is comparable to the number of required changes in the attribute description, over some class of allowed motions;

compactness: a KDS is called *compact* if the size of the certificate set it needs is close to linear in the degrees of freedom of the moving system; and

locality: a KDS is called *local* if no object participates in too many certificates; this condition makes it easier to re-estimate certificate failure times when an object changes it motion law.

KDSs have been primarily studied in the context of geometric problems that arise in virtual reality simulations. Good KDSs have been developed for a variety of spatial proximity [9, 1, 21, 17] (e.g., collision detection, closest pair, clustering), extent [4, 8] (e.g., diameter, convex hull), visibility [6, 5] (binary space partitions, occlusion), and connectivity [2, 23] (e.g., minimum spanning trees, sparse spanners) problems. For example, the three frames below from a kinetic convex hull simulation (Figure 12) illustrate a combinatorial change to the hull in a larger example, based on the algorithm described in [8]. On the bottom is a time window showing a moving ticker tape of certificate failure times. The long bars represent real changes to the convex hull, while the short bars show failures in the certification, even though the convex hull has not changed. The relative frequency of the two types of bars captures the efficiency of the KDS.

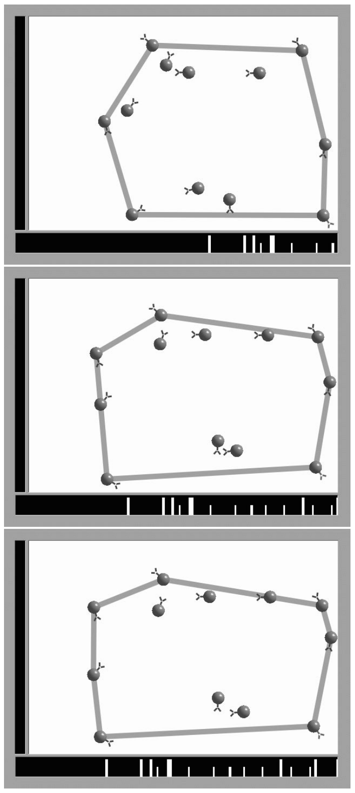


Figure 12: A combinatorial change in the convex hull.

Unlike the sensor context considered here, the virtual reality setting is significantly simpler as the instantaneous motion plans of all the objects are known and therefore reliable prediction of certificate failure times is possible. This classical KDS model needs to be augmented with considerations for the cost of sensing and communication of information, as well for dealing with uncertainty, as discussed in Sections 3 and 6.

In the context of sensor nets kinetic data structures can be used to support a variety of infrastructure tasks, including:

- maintenance of optimal sensor groups for tracking a particular moving target,
- maintenance of clusters among a set of moving vehicles; indeed the distributed clustering algorithm briefly mentioned in Section 2 can also maintain the clustering as the vehicles move, while guaranteeing nearly optimal optimality and stability for the clusters [17],
- maintenance of communication paths between friendly vehicles using the sensor net as a relay, or communication paths between nodes producing and consuming information on the net; for related work see [18].

More generally, KDSs can be used to aid in motion prediction, which is essential for successful tracking of moving targets — a good tracker always incorporates a motion predictor for the target, including both internal dynamic constraints as well as a model of global objectives. We have shown elsewhere that a KDS can be used to track the local environment of each target, e.g. the presence of obstacles, friendly or enemy vehicles, etc. [19]. This detailed knowledge of the local environment allows for much better prediction of what a target is likely to do next.

6 Probabilistic Relational Reasoning

Since noise and uncertainty is inherent in all sensing, models of both objects and their relations must be set in a probabilistic setting in order to be practically useful. For example, the belief state of the system as to the location a target t must be represented as a probability distribution (PDF). Both parametric and non-parametric representations are useful for such distributions. Though simple analytic models, like Gaussians, are the easiest to deal with, they cannot capture complex multimodal distribution shapes. The CCW example discussed in section 4 was based on representing a distribution as an array on a quantized 2-d grid of cells. A quad-tree hierarchy was implemented on top of each such target distribution to aid in the evaluation of the likelihood of the CCW relation. A popular way to proceed is to represent distributions as discrete sets of weighted *particles* in a factored position space, where each particle is a hypothesis about the position of one or a small group of objects, and its weight indicates the probability of that hypothesis. Such *particle filtering* methods have been extensively and successfully used in statistics and in several application areas [33, 26]. In its simplest form this representation assumes that the motion of different objects are uncorrelated, which is rarely true. Joint distributions can capture dependencies among the objects as needed and full or approximate factorizations can be used to reduce the dimensionality.

Particle-based representations of distributions have a number of advantages for our purposes

- they can model arbitrary distributions,
- the amount of detail can be varied as needed by controlling the number of particles used, and
- combinatorial methods can be used to deal with the computational complexity associated with large numbers of particles

We must also devise ways to represent uncertainty on the values of the complex attributes that we wish to maintain or track as the objects move. Such attributes are often geometric structures, like a Delaunay triangulation, a convex hull, or a minimum spanning tree that involve discrete (combinatorial) as well as continuous data. Thus the challenge becomes how to model distributions over such combinatorial objects. Though in in principle the particle approach can deal with arbitrary distributions, the large number of possible structures makes this approach impractical. For example, n points in the plane can have $O(n^{O(n)})$ different triangulations or spanning trees [32], each of which might be the correct attribute value. Again, the only way to deal with these exponential-size distributions is by *factorizing* them by using as many independence assumptions about the inputs as we can safely make. For example, a cycle separator in a Delaunay triangulation all of whose edges can be certified with probability 1 effectively decouples the uncertainty about what the triangulation looks like inside the cycle and outside the cycle. If we can factor a distribution into parts whose uncertainties are independent of each other, we can then use particles to represent each of the parts separately and thus attain dimensionality reduction and a great compression in the overall number of particles needed for a specified degree of approximation. Such factoring is heavily exploited in the graphical models community (Bayes nets, Markov nets) for probabilistic inference with good success [26].

Indeed *Bayesian Networks* (BNs) [27] have been a successful and widely used probabilistic model for objects when reasoning in the presence of prior beliefs and sensor evidence. Locality of interaction is embodied by explicit *conditional independence* assumptions between random variables, encoded in a graphical representation: a node is conditionally independent of all its non-descendants, given values for its parents. In the context of moving objects, *Dynamic Bayesian Networks* (DBNs) [15, 37] capture the dependence between the past and current states of an object. By appropriately enriching the notion of 'object state' we can usually make the Markovian assumption that the future state of the system is independent of its past, given the present.

While DBNs are useful models for a system whose overall composition is fixed, they are inadequate, as defined above, for the task of tracking scenes where the relations between individuals change constantly, there is uncertainty about the identity of the individuals involved [7], and the set of individuals varies over time as objects enter and leave the scene [20]. In [16, 29, 30], a new *relational* probabilistic language was introduced, which extends the Bayesian network language with additional features representing objects and the relations between them. Each object (entity) in the domain has its own attributes, which correspond to random variables in the standard Bayesian network framework. The language allows the attributes of one object to depend, probabilistically, on properties of a related object. These extended DBNs are called *Relational DBNs* (RDBNs) — they can serve as probabilistic models of relations between objects.

In our environment, the sensors and our tracking ability are rarely sufficient to determine the truth of each KDS certificate with certainty. As discussed above, a certificate fails when the sensor(s) tracking it can no longer confirm its validity with high confidence. This can occur both when the corresponding relation has become false or when the sensor(s) simply cannot verify it. Unlike the classical KDS setting where a failed certificate always leads to a proof and/or attribute repair, in our situation a failed certificate may initiate a search for alternate support structures using other high-confidence certificates, or it may have to allow for having multiple active hypotheses about the attribute value, each with some significant probability. It may also happen that no single hypothesis has significant probabilistic mass, but certain limited sets of hypotheses do. These hypotheses sets must be appropriately summarized and used as a unit by the KDS mechanism.

To ground this discussion, consider the simple certificate $a \succ b$ in the LEADER_IN_THE_CORRIDOR example above. If no camera can certify $a \succ b$ directly, and if we do not know either a or b's position with certainty, we cannot infer that $a \succ b$ with certainty. However, if we have a joint probability distribution over the (continuous) RDBN attributes representing the positions of a and b, we can determine the probability of this event. Now, consider the task of tracking the entire global relation 'a is leading the group.' Computing the probability of this event directly requires that we reason about the positions of all of the individuals simultaneously: The problem does not decouple, because our beliefs about the positions of different individuals are usually correlated. The KDS decomposes this global relation into a set of local certificates. If k certificates are used in the KDS proof, and each is true with probability $1 - \epsilon$, the total probability of error is at most $k\epsilon$.

Care, however, must be exercised in selecting the certificates. An RDBN tracking algorithm can maintain a belief state where certain correlations are maintained while others are ignored. In particular, if a and c are in different 'clusters' in the belief state, then our estimate of the joint distribution of their positions is likely to be only a rough approximation, and therefore so will our estimate of the probability that a is in front of c. Fortunately, we can integrate the strengths of the KDS framework with the RDBN belief state representation. The KDS framework allows us to 'prove' that $a \succ c$ indirectly, by having a certificate for $a \succ b$ and another for $b \succ c$. As our belief state representation allows overlapping clusters, we might have a and b in the same cluster, and b and c in the same cluster. Hence, although we might not be able to conclude directly that a is ahead of c, we might be fairly certain that a is ahead of b and b is ahead of c. (Of course, if computational resources allowed us to maintain an exact belief state about the entire situation, such local inconsistencies would not be possible.) To implement this scheme, the KDS algorithm has to be extended to search for proofs that 'match' the RDBN belief state representation. More precisely, the KDS will search for a proof based on certificates that are easy to verify with confidence based on the current belief state representation. Conversely, the KDS can guide the RDBN belief state representation, in a way that is more conducive to finding good proofs for the global properties we are trying to track.

We gave a simple example of estimating the CCW relation assuming uncorrelated target belief states in Section 4. Using reasoning such as the above, we can then proceed to attack the AM_I_SURROUNDED? scenario.

The KDS can also be used to guide the RDBNs and the their associated approximate tracking algorithms. First, the RDBN defines its dependency model via a set of relations between individuals. This set can change over time, as the situation evolves. Consider one more time our LEADER_IN_THE_CORRIDOR example, as shown in Figures 1 and 9: the KDS might become aware that a is near obstacle X and also visible from individual m (based, say, or a geometric analysis of the locations of a, m and X). This relation is used by by the RDBN to adapt its dependency model, resulting in a prediction that a will pass to the right of X so as to avoid m. As the scenario evolves, camera 3 will see a, updating our belief about a's location. The KDS will drop obstacle X from a's local environment, but add the obstacle Y; the relations used by the RDBN will be adjusted accordingly. As the relations change, the approximate tracking algorithm also changes its belief state structure to model the dependencies between the objects that interact strongly. We can allow the choice of the belief state structure to depend on the needs of the KDS, so that the belief state is more likely to provide accurate probabilities about certificates that support the global properties we are trying to track.

7 Conclusion

The sensor net architecture presented in this paper starts from a high-level description of the mission or task to be accomplished and then commands individual nodes to sense and communicate in a manner that will accomplish the desired result with attention to minimizing the computational, communication, and sensing resources that will be required. Much work remains to be done to refine and implement the relational sensing ideas presented here and validate their performance. We believe, however, that the potential pay-off for the relation-based sensing and tracking we have proposed can be large, both in terms of developing rich theories on the design and complexity of sensing algorithms, as well as in terms of the eventual impact of the deployed sensor systems.

Acknowledgments. The author wishes to acknowledge many useful discussions on the topics of this paper with Horst Haussecker, Jim Reich, and Feng Zhao at the Xerox Palo Alto Research Center, as well as Feng Xie at Stanford University. Feng Xie implmented the CCW simulations shown in Section 4. The discussion on probabilistic reasoning in Section 6 is the result of joint work with Daphne Koller of Stanford University. This research is supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-00-C-0139 to Xerox PARC, through the Sensor Information Technology Program, and in part by a grant from the Stanford Networking Research Center.

References

[1] P. Agarwal, J. Basch, L. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. In *Proc. 4rd Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[2] P. Agarwal, D. Eppstein, L. Guibas, and M. Rauch Henzinger. Parametric and kinetic minimum spanning trees. In *IEEE Symposium on Foundations of Computer Science (FOCS '98)*, pages 596–605, 1998.

[3] P. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J.E. Goodman, and R. Pollack, editors,*Advances in Discrete and Computational Geometry, in Contemporary Mathematics Series*, volume 23, pages 1–56. American Mathematical Society Press, Providence, RI, 1999.

[4] P. Agarwal, L. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving set of points. In *5-th Int. Workshop on Algorithms & Data Structures (WADS)*, pages 31–44, 1997.

[5] P. K. Agarwal, J. Erickson, and L. J. Guibas. Kinetic binary space partitions for intersecting segments and disjoint triangles. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 169–178, 1998.

[6] P. K. Agarwal, L. J. Guibas, T. Murali, and J. Vitter. Cylindrical static and kinetic binary space partitions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 39–48, 1997.

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

[8] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31:1–28, 1999.

[9] J. Basch, L. J. Guibas, and L. Zhang. Proximity problems on moving points. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 344–351, 1997.

[10] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Querying the physical world. *IEEE Personal Communi-*

cations, Special Issue on Smart Spaces and Environments, 7(5), October 2000.

[11] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *Proc. 2nd Int. Conf. on Mobile Data Management*, Hong Kong, 2001.

[12] John Byers and Gabriel Nasser. Utility-based decision-making in wireless sensor networks. In *Proc. 2000 IEEE MobiHOC Symposium*, 2000.

[13] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor queryng and routing for ad hoc heterogeneous sensor networks. Report P2001-10113, Xerox Palo Alto Research Center, Palo Alto, CA, 2001.

[14] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[15] T. Dean and M. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

[16] N. Friedman, D. Koller, and A. Pfeffer. Structured representation of complex stochastic systems. In *Proc. Fifteenth National Conference on Artificial Intelligence (AAAI)*, pages 157–164, 1998.

[17] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *ACM Symposium on Computational Geometry (SoCG '01)*, 2001.

[18] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. A maintainable spanner routing graph for ad hoc mobile networks. In *ACM Symposium on Ad Hoc Networking and Computing (MobiHoc '01)*, 2001.

[19] S. Goldenstein, M. Karavelas, D. Metaxas, L. Guibas, E. Aaron, and A. Goswami. Scalable non-linear dynamical systems for agent steering and crowd simulation. In *Proceedings of the International Conference on Robotics and Automation (ICRA '01)*, 2001.

[20] U. Grenander and M. Miller. Representations of knowledge in complex systems. *J. Royal Statistical Society B*, 56:549–603, 1994.

[21] L. Guibas, F. Xie, and L. Zhang. Kinetic collision detection: Algorithms and experiments. In *Proceedings of the International Conference on Robotics and Automation (ICRA '01)*, 2001.

[22] L. J. Guibas. Kinetic data structures — a state of the art report. In *Proc. 3rd Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 191–209, 1998.

[23] L. J. Guibas, J. Hershberger, S. Suri, and L. Zhang. Kinetic connectivity of unit disks. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 331–340, 2000.

[24] Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, SSC-2:22–26, 1966.

[25] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom '00)*, 2000.

[26] Finn Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

[27] F.V. Jensen. *Introduction to Bayesian Networks*. Springer Verlag, New York, NY, 1996.

[28] J.M. Kahn, R.H. Katz, and K.S J. Pister. Mobile networking for smart dust. In *ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom '99)*, 1999.

[29] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proc. UAI*, 1997.

[30] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proc. AAAI*, 1998.

[31] Jiři Matoušek. Geometric range searching. *Computing Surveys*, 26:421–462, 1994.

[32] J. Matoušek and J. Nešetřil. *Invitation to Discrete Mathematics*. Oxford University Press, Oxford, 1998.

[33] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[34] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *ACM Communications*, 43(5):51–58, 2000.

[35] Scott Shenker. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communication*, 13(7), September 1995.

[36] D. Sleator and E. Tarjan. Amortized efficiency of list update and paging rules. *ACM Communications*, 28(2):202–208, 1985.

[37] P. Smyth, D. Heckerman, and M.I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9(2), 1996.