CPU-friendly Tracking in Wireless Sensor Networks

Neeta Trivedi, N. Balakrishnan
Supercomputer Education and Research Center
Indian Institute of Science
Bangalore, India - 560012
neeta.trivedi@gmail.com, balki@serc.iisc.ernet.in

***Abstract*—The problem of tracking multiple targets using Sequential Monte Carlo technique under the framework of Bayesian techniques for wireless sensor networks is discussed. Distributed filtering in wireless sensor networks is an active area of research owing to the high communication costs of centralized tracking. However, distributed filtering must carefully address the conflict between high correlation among signals picked up by neighboring sensors and detached sensing by far away nodes due to limited sensing radii. Further challenges relate to the processing and communication of large number of particles in resource-constrained sensor nodes. This paper proposes a novel integrated approach to network management and target tracking by which distributed tracking is achieved in a lightweight manner. Important contributions are 'Consensus Tracking' as a low-cost distributed solution for sensor tasking and 'Multitiling' as computationally efficient solution for managing and propagating particles.**

Keywords- multitarget tracking, Bayesian filtering, sequential Monte Carlo, particle filter, sensor networks.

I. INTRODUCTION

Advancements in electronics and communication technologies coupled with research in fusion techniques have made it possible to model the non-linear and non-Gaussian nature of most real-life problems more accurately, overcoming the limitations of Kalman filter and its variants. Applications are increasingly adopting the rigorous general framework of formal Bayes modeling for dynamic state estimation problems.

Wireless Sensor Networks (WSN) has been another interesting recent development. Large networks of small untethered nodes capable of sensing, communicating and computing have opened up entirely new possibilities.

In Bayes modeling of target-tracking applications, the goal is to estimate the density $f_{k|k}(x^k \mid z^{1:k})$ of the target set being in state x^k, given all observations up to time k. The estimate is performed recursively in two steps viz. prediction and update. Prediction amounts to obtaining the prior density

$$f_{k+1|k}(x^{k+1} \mid z^{1:k}) = \int f_{k+1|k}(x^{k+1} \mid x') f_{k|k}(x' \mid z^{1:k}) dx' \quad \ldots (1)$$

The update step uses the Bayes' rule to find the posterior

$$f_{k+1|k+1}(x^{k+1} \mid z^{1:k+1}) = \frac{f_{k+1}(z^{k+1} \mid x^{k+1}) f_{k+1|k}(x^{k+1} \mid z^{1|k})}{\int f_{k+1}(z^{k+1} \mid x') f_{k+1|k}(x' \mid z^{1|k}) dx'} \quad \ldots (2)$$

Finite Set Statistics [1] extends Bayes single target tracking model to scenarios involving multiple sensors and multiple targets including target birth, death, merger and spawning by providing a unified, scientifically defensible probabilistic foundation for tracking.

The Bayes modeling problem has no closed form solution. For discrete state-spaces, exact inference is always possible, but may be computationally prohibitive [2]. A number of computationally tractable stochastic approximation techniques have been proposed. Sequential Monte Carlo (SMC) approximation or Particle filtering is one such technique that, given enough samples, guarantees to give exact answer [3]. The basic idea is to approximate the belief sate by a set of weighted particles or samples

$$f_{k|k}(x^k \mid z^{1:k}) \approx \sum_{i=1}^{N_s} w_{k|k}^i \theta(x_{k|k}^i) \quad \ldots (3)$$

for any unitless function $\theta(x)$ of a state set variable x (likewise for sets X and Z in case of multitarget tracking). The particles approach the pdf 'in asymptotia'; the larger the number of particles, the better the approximation.

Tracking in the resource-constrained WSN poses even bigger challenges. Due to the limited field of sensing of individual nodes, the set of particles must be propagated not just temporally but also spatially for computing the belief state. Centralized solutions are not suitable due to latency and excessive energy requirement. Decision on the next node responsible for tracking a target as it leaves the sensing zone of one sensor must be taken in distributed manner. Managing and propagating the particles, typically large in numbers, is highly computation and communication intensive.

This paper makes two important contributions. First, it proposes 'Consensus Tracking', a method that works in integrated mode with network management functions to designate the node responsible for tracking a target in distributed and lightweight manner. Second, it proposes 'Multitiling', a method to reduce cost of managing and propagating the particles. To the best of our knowledge, no work has been reported addressing these problems in an integrated manner.

Handling of target birth, death, merger, spawning, missed detections, false alarms, track initiation and maintenance are the scope of a related work and are not discussed here.

The rest of the paper is organized as follows. Section II discusses related prior work. Section III describes the network setting. Consensus tracking and Multitiling are

 ISBN: 978-1-61208-144-1

discussed in Sections IV and V respectively. Simulation results are shown in Section VI. Conclusions are drawn in Section VII.

II. RELATED PRIOR WORK

Many researchers have addressed the many different facets of distributed particle filtering. Liu, Chu and Reich [4] provide a survey of techniques for tracking multiple targets in distributed sensor networks. A good study on distributed target tracking is also provided in [5].

Fang, Zhao and Guibas [6] have discussed the problem of target enumeration and aggregation in sensor networks. Zhao, Liu, Liu, Guibas, and Reich [7] have considered leader-based tracking with mutual information based handing over of target to neighboring node, non-parametric storage of belief state and also a real-life implementation of distributed tracking. Coates [8] proposed two methods for reducing communication overheads: using factorization and using adaptive encoding. Särkkä, Vehtari, and Lampinen [9] proposed a Rao-Blackwellized Monte Carlo data association method in a centralized setting. The authors propose partitioning the problem into many single target tracking problem and solving the data association problem by sequential importance sampling.

Sheng, Hu, and Ramanathan [10] propose approximating the estimates with the parameters of a low-dimensional Gaussian Mixture Model (GMM). Zuo, Mehrotra, Varshney, and Mohan [11] also use GMM approximation of particles to convey belief to fusion center in bandwidth-efficient manner.

Ihler, Fisher, and Willsky [12] have considered the problem of approximating the density estimates using lossy compression techniques. Vercauteren, Guo, and Wang [13] have addressed the problem of joint tracking and classification of targets in sensor networks. Teng, Snoussi, and Richard [14] have proposed a distributed state-estimation algorithm that allows implicit compression of exchanged statistics between leader nodes.

Leader-based tracking is essential in WSN due to high correlation among signals picked up by neighboring nodes and for energy conservation. However, selecting leader node(s) as the target(s) move requires computation and communication overheads. The problem gets more complicated for multiple and unknown number of targets, and in the presence of clutter and missed detections. Realizing the importance of efficient networking infrastructure, lot of work has been carried out in this area (reference [15] provides a review); however, almost all of the 'tracking leader election' is independent of such structures. Also, while it is important to reduce communication cost, which is order of magnitude higher compared to computation cost, the energy consumption due to computation cannot be ignored, more so given the typically large number of particles in tracking problems. To the best of our knowledge, no work has been reported that addresses computation cost.

III. NETWORK SETTING

As discussed by Sheng, Hu, and Ramanathan [10], the assumption of independent observations at individual sensor nodes is unrealistic, and hence the motivation to partition the network into cliques. The clique size proposed in [10] is enough to cater to the spatio-temporal bandwidth of signals. As the target moves, newer cliques are formed. The authors argue that it results in unpredictable and time-varying size of clusters. In such cases, the cost of network management is typically very high and tends to dominate the operations cost. The cost of creating newer clusters every time results in additional overheads [16]. Most importantly, when multiple targets move in a given region, it is non-trivial, even impossible in some cases, to form distinct cliques for individual targets. Hence it is ideally suited that network be partitioned efficiently, not dependent on target movements, and sensor tasking be done using a robust distributed mechanism.

N. Trivedi and N. Balakrishnan have earlier proposed *iGroup* [16], a lightweight distributed algorithm to partition the sensor network into hexagonal cells and also to decide on unique communication channels and time slots for interference-free inter- and intra-cell communication. In communication infrastructure, the number of neighbors being bounded by six allows pre-determination of communication channels and time slots in a way that avoids interference. Figures 1 and 2 indicating the organization, channel usage and time slot allocation have been reproduced from [16] for reference.

The numbers inside cells in Figure 1 indicate channel indices. Many mote systems allow selection of channel from a set of tunable frequencies up to 25 in number, and the channel can be changed at run time simply by using a system call by providing the index as parameter. In Figure 2, interpretation of data can be determined by the control messages. The cell diameter is chosen such that the neighboring cells can communicate using the available power levels, and that the desired sensing coverage can be obtained when only the cell leaders are awake.

This paper demonstrates how this infrastructure can be exploited for efficiency in tracking problems. Though discussed in this setting, 'Consensus Tracking' can be used with other networking structures with ease.

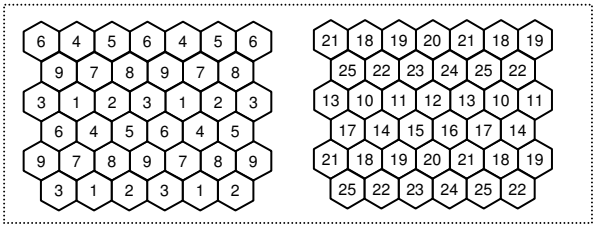

Figure 1. Intra-cell (left) and Inter-cell (right) channel usage patterns

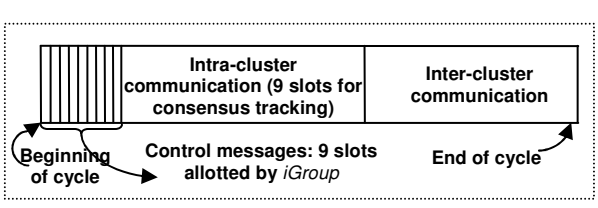

Figure 2. Network Operation Cycle (modified for Consensus Tracking)

IV. CONSENSUS TRACKING

To exploit spatial diversity, multiple nodes observe a particular target; however, in a hierarchical infrastructure, it is the cell leader's responsibility to track target(s) using detection details from itself and the other cell members. Two neighboring cells may detect the same target(s) and hence it is required to decide which cells must track which targets.

The theme of Consensus Tracking is that the cell that is assumed to contain a given target must track it. Of course, the target position is never known in advance and hence a distributed mechanism is required by which all neighboring cells can uniquely agree on an assignment. This is achieved as follows.

All members belonging to the cell having intra-cell channel ID '*m*' broadcast their findings in slot '*m*' on channel '*m*' using transmission power enough to reach the neighboring cells. Random backoff is used to avoid collision. Having received the signals from own cell members and the neighbors', each cell generates 'energy plots' [5]. Figure 3 shows a sample energy plot. It is assumed that the sensor density is at least enough to capture the peaks and valleys of the energy field. The peaks in this plot indicate target positions. The cell to which the peak belongs is responsible for tracking that target. Since the same transmission is heard by all the neighbors for common targets and same algorithm used for generating energy plots, the peaks are common (error handling is discussed in the following subsection). The distributed algorithm for checking cell boundary [16] ensures that the cell assignment is also unique. Due to the properties of network infrastructure created by *iGroup*, neighbors more than one hop away need not participate in the coordination.

The assignment of targets to cells is followed by decision on whether to hand over the history. Specifically, if one cell was tracking a target that has just crossed its boundary, the current cell must hand over the time-series measurement data to the neighbor. The tracking process *per se* is out of scope of this paper; however, Section IV discusses computationally efficient methods towards distributed state-estimation process as well as towards preparing for handover.

A. Effect of Message Errors

If the presence of transmission or reception errors, the neighbors may end up with different peaks and sensor tasking may be ambiguous. However, the Bayesian tracking framework is robust enough to accommodate process and measurement noise and, as the further analysis and simulation results show, this fluctuation gets smoothened out in the tracking framework.

The framework for Consensus Tracking consists of an outer loop responsible for considering target births and deaths including clutter handling. Targets are identified by a grid-identification mechanism. When a target is detected for the first time, it is associated with a 'time-to-live' counter. If it was a false detection, this counter will eventually expire and the track will be deleted. Missed detections are handled in a similar manner by letting the tracks live for a 'time-to-live' counter. This mechanism helps combat occasional message errors as discussed below.

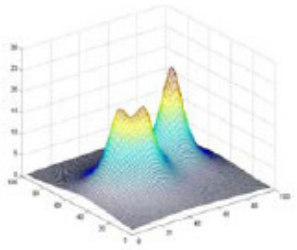

Figure 3. Sample energy plot [5]

Detection or transmission errors could lead to a target being detected in wrong position by all relevant neighboring cells. As a result, a wrong cell may begin tracking it. If the target was an existing one originally belonging to some other cell, the error would lead to missed detection there. Assuming that errors are spurious and non-identical over different time periods, the subsequent cycles will be able to recover from this error, where one cell assumes the phenomenon to be a false alarm and the other treats it as missed detection.

Reception errors could lead to generation of different peaks by neighboring cells and as a result, a target may be detected in two different positions by neighboring cells, potentially both of which may be wrong. Even this will be treated as false alarm and/or missed detections and subsequent cycles will allow recovery from this error as well.

A target moving on the cell boundary for some time could potentially cause fluctuating handovers. To avoid this, a cell hands over tracking to a neighbor only when the target is consistently in the other cell boundary for at least 3 cycles. The neighboring cell is informed not to initiate track during this period.

V. MULTITILING

Computation steps in SMC techniques involve particle initialization, prediction, assigning importance weights based on measurement and sequential importance sampling to alleviate impoverishment. A large number of particles must be used if the tracking has to be accurate. This involves large number of computationally expensive floating point operations. Multitiling is a low-cost solution to this problem.

A. Initialization

In the absence of any prior knowledge, the initial measurement is best representation of the density for the belief state. The number of peaks is representative of the number of targets initially present in the system. Assuming typical sound source point target at location ζ with amplitude a and lossless, isotropic sound propagation model, the root-mean squared amplitude measurement z at location x is given by

$$z = \frac{a}{\| x - \zeta \|} + w \qquad \ldots (3)$$

where w is measurement noise. The measurement function is discretized with desired resolution to generate particles.

B. Prediction

State transition is given by

$$x_{t+1}^i = g(\theta_{t+1}^i)x_t^i + w_t^i \quad \ldots (4)$$

where w is the process noise. The first prediction typically assumes a large variance for process noise for each state vector component. The variance gets reduced through the predict-measure-update cycle.

C. Update

Measurement is given by

$$z_t^i = h^i(\theta_t^i)x_t^i + v_t^i \quad \ldots (5)$$

where v is measurement noise. The distance between $z_t^i = h^i(\theta_t^i)x_t^i + v_t^i$ and $z'^i_{t|t-1} = h^i(\theta_{t|t-1}^i)x'^i_{t|t-1}$ is the prediction error assuming $z'^i_{t|t-1}$ is what measurement would have been if state were $x'^i_{t|t-1}$ as predicted at time t-1.

Importance weights to be assigned to particles are inversely proportional to the 'distance' between predicted and actual measurements. In the cases where state vector involves non-spherically symmetric distributions, mere Euclidean distance is not a good metric. Distance measure that takes into account the variance of the variables is required. This can be achieved by scaling the variables by their 'variability'. Mahalanobis distance is one such measure that does not just take this variability into account but also caters to covariance between variables. Mahalanobis distance between two vectors $x = (x_1, x_2, ..., x_N)$ and $y = (y_1, y_2, ..., y_N)$ in $\Re^N$ is defined as

$$D = \sqrt{(x-y)'\Sigma^{-1}(x-y)} \quad \ldots (6)$$

where Σ is the covariance matrix of the distribution.

Mahalanobis distance must be calculated for each particle during every predict-update cycle for all the targets. The operation requires many floating-point multiplications. Given that the number of particles is typically very large, this is a highly computation-intensive step. In the following section the authors propose a low-cost approximation to this step. As the approximation progresses, side information is stored that helps in quick consolidation of state-estimate to be exchanged between cells either for belief handover or for distributed state estimate.

1) Tile-based Weight Assignments

The particles that are 'closer' to the measurement must be weighted higher. Points equidistant (in Mahalanobis sense) from the center form an ellipsoid around the center. A less computation-intensive approximation would be to consider cuboids enclosing the ellipsoid boundaries, because then the 'distance' could be calculated by only comparing the values with minimum and maximum values. This is the principle behind Multitiling. The region of interest in $\Re^N$ is divided into bigger cuboids enclosing smaller ones having common center point; this center point is the measured target state vector. Distance between the cuboids could be fixed for a fixed-interval comparison, or could increase with distance from the center for variable-intervals. Particles inside one cuboid share common weight.

Spatial data structures such as k-D tree are ideally suited for proximity-based searches even when state vector contains elements other than position [12]; however, the weight assignment process requires the entire particle set to be processed i.e., all the nodes in the tree must be visited. This results in undue space and time complexity compared to array-based representations. Nevertheless, the distance comparison concept in k-D tree is computationally efficient and is used in Multitiling for assigning weights. Multitiling treats concentric ellipsoids as 'bins', with particles in one bin considered to have equal weights. These bins approximated as cuboids allow simpler boundary comparisons.

a) Statistically Independent State Variables

When the covariance matrix is diagonal, i.e., the different state variables are statistically uncorrelated, the axes of the ellipsoids are parallel to the principle axes of $\Re^N$. Without loss of generalization, consider $\Re^2$. The area of ellipse is $\pi*A*B$ where A and B are the axes of the ellipse. The area of the enclosing rectangle is $4*A*B$. Scaling down the axes of the ellipse i.e., the bin sizes by 0.9 ensures that the rectangle covers desired area. Memberships get affected only for those particles on the boundary, which is an acceptable approximation considering the savings in computation. Figure 4(a) depicts this approximation called Multitiling-Z.

The test for a particle to lie within given cuboid boundaries involves simple order comparisons. Taking an example of 2-D Euclidean space, equidistant points form a circle and a low-cost approximation requires checking for (x < *center_x*+radius) and (y < *center_y*+radius). Extending this approximation to non-Euclidean distances means checking for ($\acute{\alpha}$ < *center_*$\acute{\alpha}$ + $\acute{\alpha}$*_radius*) and ($\acute{\varepsilon}$ < *center_*$\acute{\varepsilon}$ + $\acute{\varepsilon}$*_radius*), where $\acute{\alpha}$*_radius* and $\acute{\varepsilon}$*_radius* are the scaled radii of the ellipse corresponding to variances in $\acute{\alpha}$ and $\acute{\varepsilon}$ respectively.

The range for each variable in the state vector is divided into intervals that are multiples of σ_s (variance or spread) for that variable. In the present work, ranges are considered as multiples of $0.5\sigma_s$ for each variable.

b) Correlated State Variables

Some elements of the state vector may be correlated with each other, causing the concentric ellipsoids that depict 'scaled equal distances' to be inclined at a non-zero angle with respect to the principle axes in $\Re^N$. Mahalanobis distance calculation uses covariance for scaling. The inclined axes can be projected to the principle axes, effectively uncorrelating the elements; however, comparisons in this case will involve first projecting the particle vectors to the new coordinate system, involving costly operations. The following simplification by fitting ellipsoids to cuboids called Multitiling-N is proposed. $\Re^2$ is considered for illustration without loss of generality.

As a preprocessing step, a rotated rectangle is fitted to the ellipse (Figure 4(b)). It is required to compute the projection

on each axis in $\Re^2$ only once; thereafter additions and subtractions can be used for finding the corner points of the rectangle. An acceptable quantization factor is decided, which is used for approximating the rectangle edges by stair cases. Even this operation uses only additions and subtractions. The resulting quantization effect is a good tradeoff towards the gain in speed and closeness of approximation. Figure 5 depicts the staircase approximation.

Having preprocessed the (quantized) edge points, Multitiling comparisons in each cycle amount to finding the staircase step in one direction followed by checking the boundary in the other direction. It can be easily seen that there are only two points on the horizontal axis for one step on the vertical axis. The number of comparisons during the update cycle is identical to that in the case of Multitiling-Z.

2) Joint Tracking

Multitarget tracking can be implemented as multiple independent particle sets or as multitarget particle systems, the latter being far more complex because it must deal with state sets rather than state vectors. The former case benefits directly from the approximations of the previous section. Multitiling concept applies equally well to the joint tracking case, though it can look a bit cumbersome.

It is convenient to order [1] $X^0_{k|k}, X^1_{k|k},..., X^{\upsilon}_{k|k}$ by increasing target number i.e., $w^0_{k|k}$ is the weight of the no-target particle $X^0_{k|k} = \phi$, the next υ_1 particles $X^0_{k|k}, X^1_{k|k},...X^{\upsilon_1}_{k|k}$ represent single-target samples $X^0_{k|k} = \{x^1_1\},..., X^{\upsilon_1}_{k|k} = \{x^{\upsilon_1}_1\}$ and, in general, the n-target states are represented by υ_n multitarget particles, up to some largest number ň of particles. Thus $\upsilon = 1 + \upsilon_1 + \upsilon_2 + ... + \upsilon_n$, where υ_0=1 is the number of zero-target particles.

Define $\theta(X)$ as $\theta(X)$ =1 if | X |=n and $\theta(X)$ =0 otherwise. Then, $f_{k|k}(n \mid Z^{(k)}) \cong \sum_{i=\upsilon_{n-1}+1}^{\upsilon_n} w^i_{k|k}$, where $f_{k|k}(n \mid Z^{(k)})$ is the cardinality distribution of $f_{k|k}(X \mid Z^{(k)})$ i.e., $f_{k|k}(n \mid Z^{(k)})$ is the probability that n targets are present in the scene [1].

Multitiling addresses joint tracking as follows. The number of peaks is counted and number of targets *n* estimated from there taking into account the probabilities of missed detection and false alarms. The sets having *n* targets are accordingly assigned weights. Probabilities of missed detection and false alarms indicate possible variance, and the sets having *n-l* to *n+l* targets are assigned reducing weights accordingly.

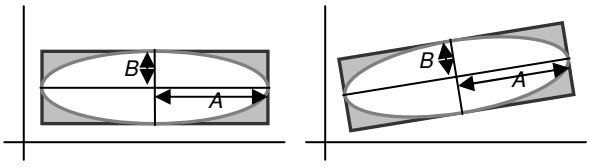

Figure 4. The distribution 'ellipse' for 2-D state vector when the two dimensions are (a) independent (Multitiling-Z) (b) having non-zero correlation (Multitiling-N)

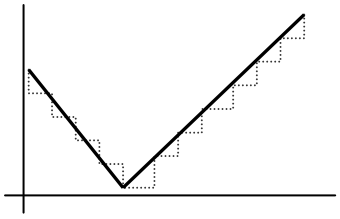

Figure 5. Quantizing rectangle edges for Multitiling-N

Mahalanobis distance must now be calculated for each particle in the set for permutations of locations. The weights assigned to the cardinality multiplied by the weights to the particle set by Multitiling are the net weights for the sets.

3) Computational Complexity

The computation complexity of Multitiling is order of magnitude less compared to direct method of distance calculation for the same number of state variables. Mahalanobis distance is calculated as $D = \sqrt{(x-y)'\Sigma^{-1}(x-y)}$. Even if we were to ignore the square root (only ordering is needed, not the absolute distance) and the inverse of Σ (assuming known covariance matrix, it is enough to calculate the inverse once), it requires O(N) floating-point subtractions and O(N^2) floating-point multiplications.

In contrast, Multitiling-Z requires O(N) floating-point comparisons only. Multitiling-N additionally requires preprocessing. Since the distribution covariance is known, the computation-intensive operations such as fitting the cuboids can be done offline and fed to the sensor nodes.

In the O(N) comparisons, the constant factor depends on the number and spread of bins (Б) and, in the case of Multitiling-N, the desired quantization factor (Ќ). Assuming that Б<<ν (number of particles) and the number of steps due to Ќ<<ν, the cost of binning and quantization does not dominate. This is huge saving considering the typical number of particles to be handled in each cycle of real-time operations. The saving is even more significant in the case of joint tracking, where distances for many permutations of target states must be computed.

4) Resampling

Since the particles have unequal weights, they must be replaced by new particles having equal weights while retaining the influence of weights. The weights assigned by Multitiling are analogous to those assigned by existing methods and hence sequential importance sampling for Multitiling can be performed using common methods. This paper uses multinomial resampling. Assuming ν particles and $\bar{w}_i \equiv w^i_{k+1|k+1}$ for all i=1…ν, the multinomial distribution $\mu(i_1,... i_\nu) = \frac{\nu!}{i_1!... i_\nu!} . \bar{w}_1^{i_1} ... \bar{w}_\nu^{i_\nu}$ is a probability distribution on all ν-tuples (i_1, …, i_ν) of nonnegative integers for which i_1 + …+ i_ν = ν. A random sample (e_1, …, e_ν)~μ(.) is drawn from this distribution. If e_i=0, then the particle $x^i_{k+1|k+1}$ is

eliminated, else e_i identical copies are made of this particle. Since $e_1 + \ldots + e_\nu = \nu$, total number of particles stays as ν.

To avoid particle impoverishment, the copies of particles are randomly jittered.

5) Belief Handover

Belief propagation always requires more bits than raw data and hence is extremely expensive. Gaussian Mixture Model (GMM) approximation is an excellent way to save the transmission cost [10][11]. However, there usually is no prior knowledge of the number of components 'c' in the mixture. The μs and σs are also not known in advance. Reference [10] suggests selecting c as a function of number of targets and learning the GMM parameters using an iterating Expectation-Maximization (EM) algorithm. Reference [11] picks up c on a higher side subject to a maximum number, merges two components if the Kullback-Lieber distance between them is below a threshold, and applies EM algorithm.

De facto method for initialization of EM has been to assign random values to the parameters. However, EM is guaranteed only to converge to local optimum in the likelihood and hence it is best to initialize the model in the region of likelihood space where the local maxima are supposed to lie. Multitiling provides a crucial cue to c, μ, and σ, arguably the closest approximation for EM iterations, thereby saving precious iteration cost and also resulting in more accurate mixture parameters. The cue is provided as follows.

A mode is a local maximum and is represented by higher frequency of particles. In Multitiling, finding local maxima amounts to counting the number of particles in each bin. Multitiling divides the particles into bins having equal 'weighted' distance from the estimated state; however, selection of mode requires both range and bearing in the multidimensional space. First, the bins corresponding to local maxima are identified based on first-order differences; threshold is applied to avoid spurious small maxima. The number of maxima thus found is indicative of c. One random particle each from these bins is assumed to be the mean 'μ' for the mixture component. The variance 'σ' is estimated based on the first order difference.

This process does not take into account the possibility of multiple modes for a single target in a single bin e.g., equal probability of a target taking a left or a right turn, indicated by two modes on the opposite sides. To address this, Multitiling maintains another variable called 'quadrant' for each particle. Orthogonal hyperplanes in $\Re^N$ divide the cuboids into four 'quadrants'. No additional cost is needed to find this quarter through Multitiling process. Binning is done separately for the quadrants. Modes closer than this are considered as data fluctuations and are filtered out during thresholding.

VI. SIMULATION

An area of 100mx100m is considered with 400 nodes forming the *iGroup* infrastructure, and a target is moved through the area.

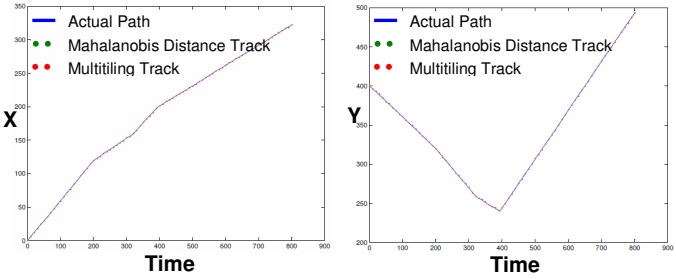

Figure 6. Tracking using Multitiling-Z

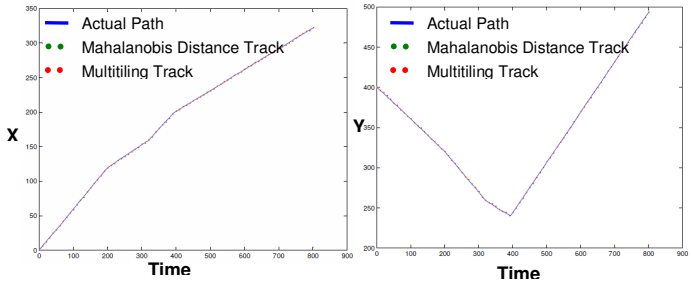

Figure 7. Tracking using Multitiling-Z: Zoomed-in version

A. Consensus Tracking

The metrics considered are multiple tasking (assigning one target to 2 or more cells), incorrect tasking and handover fluctuations (at least once). Results are averaged over 20 simulation runs. Parameters as given in Table-I are assumed, where P_d is probability of detection and λ is probability of false alarm. Results are shown in Table-II. Note that due to sensors erring in estimating their own location, a target in one cell could appear to be in another cell. This is not considered to be an error in target assignment.

TABLE I. PARAMETERS FOR CONSENSUS TRACKING

Scenario	1	2	3	4	5	6
P_d	1	0.9	1	1	1	0.9
λ	0	0	0.001	0	0	0.001
Transmission error	0	0	0	0.01	0	0.01
Reception Error	0	0	0	0	0.01	0.01

TABLE II. CONSENSUS TRACKING: RESULTS

Scenario	1	2	3	4	5	6
Multiple Tasking	0%	0%	0%	0%	<1%	<5%
Incorrect Tasking	0%	0%	0%	0%	<1%	<5%
Handover Fluctuations	0%	0%	0%	0%	0%	0%
Recovery after Error	NA	100%	100%	100%	100%	100%

B. Multitiling-Z

Figure 6 shows the actual (with process noise), Mahalanobis-estimated, and Multitiling-Z-estimated X and Y positions. Figure 7 shows a smaller sector zoomed in for clarity. Results were averaged over 20 simulation runs.

 ISBN: 978-1-61208-144-1

C. *Multitiling-N*

A hypothetical ellipse like the one in Figure 4 (b) is considered with variances and correlations as listed in Table-III. Monte Carlo simulations were used to generate data points with these statistical parameters; these points represent particles. State estimates were computed using Mahalanobis distance and Multitiling-N approximation with various quantization levels. The estimation errors in both cases are depicted on the scatter plot of Figure 8.

TABLE III. PARAMETERS CONSIDERED FOR MULTITILING-N

σ_x^2	σ_y^2	σ_{xy} (Covar)	$\rho_{x,y}$ (Correlation)
16	25	[10, -7, 0, 0.5]	[0.5, -0.35, 0, 0.025]
25	9	[-8, 13, 0, 0.9]	[-0.53, 0.87, 0, 0.06]
64	49	[50, -7, 0, 0.1]	[0.89, 0.125, 0, 0.0018]
49	49	[24, -12, 0, 0.8]	[0.49, -0.25, 0, 0.016]

VII. CONCLUSION AND FUTURE WORK

Using the strength of the Bayesian technique atop the foundation of a robust networking infrastructure helps build a robust distributed tracking framework. Use of Multitiling provides the additional cost-saving.

Results in Table-II will change for multi-target scenario, especially when targets move in close proximity. Handling target energy overlap is the subject of a related work and hence not discussed here; however, multi-target tracking can only build out of robust single target tracking.

Multitiling provides excellent approximation for Mahalanobis distance, and at a substantially reduced cost. Even in the case of Multitiling-N (Figure 8), the worst-case estimation error is less than 3 units (most of the worst-case numbers belong to the cases of large variances and coarse quantization), whereas most other errors are confined to ±0.5 units, similar to what Mahalanobis distance based estimates provided. The framework makes a low-cost tracking model.

Our ongoing work is on implementation of EM and using it with data generated by Multitiling to demonstrate increase in performance and reduction in iterations.

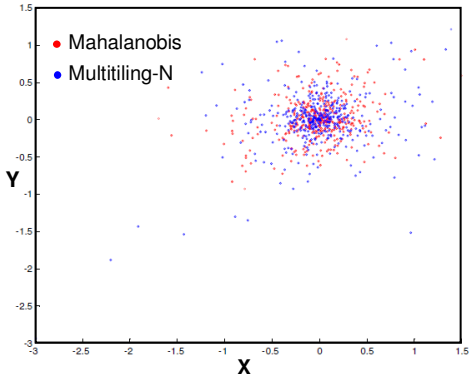

Figure 8. State Estimation Errors using Mahalanobis Distance and Multitiling-N

REFERENCES

[1] R. P. S. Mehlar, Statistical Multisource-multitarget Information Fusion. Artech House, Norwood, MA: 2007, ISBN 13: 978-1-59693-092-6

[2] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning", PhD Thesis University of California, Berkeley, Fall 2002

[3] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-Gaussian Bayesian state estimation", in Proc IEEE, vol. 140, pp. 107-113, 1993

[4] J. Liu, M. Chu, and J. E. Reich, "Multitarget Tracking in Distributed Sensor Networks", IEEE Signal Processing Magazine, Vol. 24, Issue 3, May 2007, pp 36-46, doi 10.1109/MSP.2007.361600

[5] M. E. Liggins, C.-Y. Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos, "Distributed Fusion Architectures and Algorithms for Target Tracking", Proc. IEEE, vol. 85, Issue 1. pp. 95-107, Jan 1997, doi: 10.1109/JPROC.1997.554211.

[6] Q. Fang, F. Zhao, and L. Guibas, "Lightweight Sensing and Enumeration Protocols for Target Enumeration and Aggregation", In Proc. 4th ACM Intl Symp Mobile Ad Hoc Networking and Computing (MobiHoc'03), 2003, doi: 10.1145/778415.778436

[7] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative Signal and Information Processing: An Information Directed Approach", Proc. IEEE, Aug 2003, Vol. 91, Issue 8, pp 1199-1209, doi: 10.1109/JPROC.2003.814921

[8] M. Coates, "Distributed Particle Filters for Sensor Networks", In Proc. 3rd Intl Symp Information Processing in Sensor Networks (ISPN'04), 2004, pp. , doi: 10.1145/984622.984637

[9] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized Monte Carlo Data Association for Multiple Target Tracking", In Proc. 7th Intl Conf. Information Fusion, July 2004, pp 583-590

[10] X. Scheng, Y.-H. Hu, and P. Ramanathan, "Distributed Particle Filter with GMM Approximation for Multiple Target Localization and Tracking in Wireless Sensor Networks", In Proc. 4th Intl Symp Information Processing in Sensor Networks, Apr 2005, pp 181-188, doi: 10.1109/IPSN.2005.1440923

[11] L. Zuo, K. Mehrotra, P. K. Varshney, and C. K. Mohan, "Bandwidth-efficient Target Tracking in Distributed Sensor Networks using Particle Filters", In Proc 9th Intl Conf Information Fusion, July 2006, pp 1-4, doi: 10.1109/ICIF.2006.301692

[12] A. T. Ihler, J. W. Fisher, and A. S. Willsky, "Particle Filtering under Communication Constraints", In Proc. 13th IEEE/SP Intl. Workshop Statistical Signal Processing, July 2005, pp. 89-94, doi: 10.1109/SSP.2005.1628570

[13] T. Vercauteren, D. Gou, and X. Wang, "Joint Multiple Target Tracking and Classification in Commaborative Sensor Networks", IEEE J. Selected Areas in Communications, Vol. 23, Issue 4, pp. 714-723, Apr 2005, doi: 10.1109/JSAC.2005.843540

[14] J. Teng, H. Snoussi, and C. Richard, "Binary Variational Filtering for Target Tracking in Sensor Networks", In Proc. 14th IEEE/SP Intl. Workshop Statistical Signal Processing, Aug 2007, doi: 10.1109/SSP.2007.4301346

[15] N. Trivedi, S. S. Iyengar, and N. Balakrishnan, "Ripples: Message-efficient, Coverage-aware Clustering in Wireless Sensor and Actor Networks", Intl J. Communication Networks and Distributed Systems", Vol. 2, Issue 1, Jan 2009, doi: 10.1504/IJCNDS.2009.021697

[16] N. Trivedi, S. S. Iyengar, and N. Balakrishnan, "Efficient Multiplexing for Multichannel Data Dissemination with Delay Guarantees in Wireless Sensor Networks", In Proc. 2nd Intl Conf. Sensor Technologies and Applications (SENSORCOMM-08), Aug 2008, pp 23-29, doi: 10.1109:/SENSORCOMM.2008.109

 ISBN: 978-1-61208-144-1