PARALLEL PARTICLE FILTERS FOR TRACKING IN WIRELESS SENSOR NETWORKS

Garrick Ing and Mark J. Coates

McGill University
Department of Electrical and Computer Engineering
3480 University St., Montreal, Quebec, Canada, H3A 2A7

ABSTRACT

An important application of wireless sensor networks is the tracking of objects moving through a monitored area. In some circumstances, a particle filter can perform substantially better than other tracking algorithms. A simple implementation is to transmit measurement data gathered at distributed sensors to a fusion centre and apply a single particle filter. The filter estimates the current position and predicts future locations so that appropriate sensors can be activated. This centralized approach can be energy-expensive and prone to failure: uncompressed data must be transmitted across multiple hops and there is a concentration of data transmission around the fusion site, which constitutes a single point of failure. This paper addresses these issues by proposing a distributed particle filter implementation in which parallel filters run at multiple nodes. These shared filters are used to quantize vectors of measurements. Simulations indicate that the scheme significantly reduces the energy expenditure of communication.

1. INTRODUCTION

One of the primary considerations when performing object tracking using a sensor network is how to maintain estimation accuracy whilst maximizing network lifetime (by minimizing energy consumption). One approach is to collect the distributed measurements at a central site and apply a tracking algorithm. Typical algorithms used when facing non-linear dynamics and/or non-Gaussian observation noise are the extended Kalman Filter (EKF) [1, 2], Gaussian sum approximations [3] and grid-based filters [4]. Sequential Monte Carlo methods, such as the particle filter (in all its guises) [5], can perform substantially better than these other tracking algorithms; the penalty is that they are generally more computationally demanding [6].

A particle filter maintains a set of "particles" that are simply candidate state values. The filter evaluates how well individual particles correspond to the dynamic model and set of observations, and forms a state estimate through an appropriate weighted averaging of particle values (or perhaps a maximization). Most tracking algorithms using particle filters in sensor networks adopt a centralized approach; the particle filter resides on one sensor node (although this *leader* node may change over time) [7]. This approach has several disadvantages. Centralization introduces a single point of failure and can lead to high, unevenly distributed energy consumption because of the heavy communication cost involved in transmitting the data to the fusion centre.

Distributed algorithms, such as the distributed particle filter (DPF) algorithm proposed in [8], address the aforementioned problems. These algorithms decentralize the computation or communication so that a single fusion centre is not required. The DPF works by maintaining a local particle filter at selected nodes throughout the network. Each local filter is used both to perform estimation and to implement extensive compression of local measurements for transmission to other nodes. The DPF, while mitigating some of the inherent problems of centralization, can also be computationally and communicationally expensive if the communication is not handled efficiently. This paper proposes two improvements to this distributed particle filter algorithm to reduce communication overhead. First, we describe an improved quantization and encoding step that performs Huffman coding by constructing the tree from the particle filter's information. Second, we introduce a vectorization scheme that reduces the fraction of communication energy wasted through transmitting packet headers.

The remainder of the paper is organized as follows. Section 2 discusses related work, and Section 3 reviews the distributed particle filter of [8]. Section 4 describes the proposed parallel distributed particle filter, and Section 5 analyzes its performance. Section 6 provides concluding remarks.

2. RELATED WORK

Several approaches for performing decentralized object tracking in sensor networks have been proposed in the literature. In [7], a "leader node" concept is used where one node is selected to track the object at every time step using a particle filter. A hand-off of information to a new leader node is therefore required whenever the leader changes. This in-

volves transmitting the particle filter in the form of raw particle values and weights or training and communicating a parametric approximation. The authors of [9] adopt a similar approach to the DPF of [8] but use an EM algorithm to train a Gaussian mixture approximation to the particle representation, and then exchange the mixture parameters. In [10], each node has its own local particle filter. Using a query-response system, neighbours exchange the entire state trajectory of a small subset of the particles (those with highest weight). Shin et al. [11] proposed a distributed method whereby nodes use their local data to formulate an estimate of the state and only transmit data when other users of the system request them.

3. DISTRIBUTED PARTICLE FILTER

The distributed particle filter (DPF) algorithm of [8] maintains particle filters at a set of nodes dispersed throughout the network. The sensor network architecture consists of two different types of nodes. Class B nodes (or sensors), when active, are responsible for taking measurements related to the object's position. Class A nodes, which have more processing power and energy, are responsible for running the particle filters to track the object. Each class A node is associated with a set of "children" class B nodes and collects data from these children, if any are active.

The unique aspect of the DPF is that the local particle filter at each active class A node is used to efficiently encode the local measurement. In order to encode data at measurement instant $t + 1$, the local particle filter at time t is propagated (blindly) according to the dynamic model. Each particle then has an expected measurement value for a given sensor and these expected values are used to quantize the actual measurement using a Lloyd-Max algorithm. Straightforward labelling of the data bins is then used to encode the local data for transmission to the other class A nodes. In order for this quantization and encoding scheme to be successful, it is crucial that all local particle filters match in the sense that the set of particles are identical. This can be achieved by initializing the particle filters using the same random seed and ensuring that they all propagate based on the same distributed, quantized measurements. Note that all class A nodes must have some knowledge about the measurement process at each sensor node, for example, sensor position and calibration. This information can be communicated with the encoded data if necessary.

The distributed particle filter algorithm works in the following manner, which is repeated at every time step: 1. Selected class B nodes (located close to the predicted position of the object) collect measurements related to the object's state. 2. The measurements from each active class B node are passed to its associated class A node. 3. The particles on each active class A node are propagated using the state dynamic model and a histogram of the expected measurement values is constructed. 4. The measurements are quantized using the Lloyd-Max algorithm and encoded by bin labelling. The encoded values are transmitted to all other class A nodes. 6. Each class A node decodes all the encoded transmission to obtain the quantized measurement values. 7. Each particle filter acts using these quantized measurements to generate an estimate of the object's state.

4. PARALLEL DISTRIBUTED PARTICLE FILTER

In this paper, we propose two modifications to the distributed particle filter to reduce communication costs: a new quantization/encoding procedure step and a vectorization step. We call our proposed algorithm the *Parallel Distributed Particle Filter* (PDPF).

4.1. Quantization and Encoding

Data transmitted through a sensor network is generally quantized to some extent; the simplest form of quantization is probably accomplished by dividing the entire measurement space into small regions and transmitting the label of the region where the data lies. However, this scheme may still require a substantial number of bits to represent the data. The DPF in [8] uses a Lloyd-Max algorithm trained according to propagated particle filters to quantize the measurement and achieves a much higher compression ratio, but the Lloyd-Max algorithm is computationally expensive. Here we use a much simpler quantization method but employ an efficient encoding scheme to achieve similar compression.

The quantization process commences by blindly propagating the particle filter from the previous time instant and calculating the expected measurement for each propagated particle. We then divide the range of the expected measurements into equal bins and form a histogram. These steps are illustrated in Figure 1. We construct a Huffman tree [12] using the histogram to develop the codebook used to encode the data. This measurements can then be encoded by using the Huffman tree codeword representing the bin associated with the data. If the propagated particles are a good representation of the state, then the measurement should lie in a densely-populated bin and the codeword will consist of very few bits. In the decoding step, the quantized measurement values can be reconstructed by recreating the same Huffman tree because each class A node has a copy of the same particle representation.

4.2. Vectorization

In most cases, during any time instant, only one or two class A nodes collect measurements. Compression of these measurements can result in the need to transmit only 10-15 bits. Since a packet header in most transmission systems

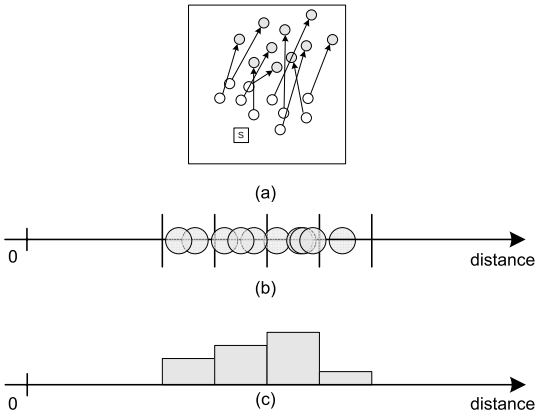

Fig. 1. Example of quantization/encoding a specific class A node. (a) Original particles (circles) are propagated using the dynamic model. Shaded circles are the propagated particles. (b) The expected measurements of the propagated particles are split into bins of equal size. (c) A histogram of the number of particles in each bin is constructed.

require at least 6 bytes, much of the communication energy is wasted on transmitting overhead bits. In this section we describe a procedure for delaying the broadcast of the encoded measurements until several time steps have elapsed. We call this scheme *vectorization*.

As an example, suppose each class A node collects data for five time steps before distributing its vector of compressed values to the other nodes. During the intervening time steps, only the active class A nodes will have up-to-date *local* estimates and these will vary between nodes, as they are formed using only local measurements. The quantization of the data is performed using a blindly propagated particle filter. When the five time instances have elapsed, the vector of encoded measurements is shared with other class A nodes and they run their local particle filter on the global quantized data. At this point, a *global* estimate of the state can be formed.

The scheme achieves a substantial communication saving by reducing the ratio of overhead bits to data bits. For a given compression ratio, the accuracy of the global estimates diminishes because the particle filters used for encoding the latter measurements in the vector are propagated farther into the future and become a less reliable indicator of the expected values. If estimates are required at every time step, then local estimates must be used – we use an average of the estimates at the active class A nodes. The local estimates are less accurate than the global estimates because they are formed from a subset of the measurements.

4.3. PDPF Algorithm

We consider the problem of estimating the current state of an object using a Markovian state-space model that is (potentially) non-linear and non-Gaussian. The unobserved state $\{\mathbf{x}_t; t \in \mathbb{N}\}$ is modelled as a Markov process with initial distribution $p(\mathbf{x}_0)$ and a transition probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The observations $\{\mathbf{y}_t; t \in \mathbb{N}^*\}$ are assumed to be conditionally independent in time given the process $\mathbf{x}_t$ and of marginal distribution $p(\mathbf{y}_t|\mathbf{x}_t)$. The system state and observations up to time t is denoted by $\mathbf{x}_{0:t} \triangleq \{\mathbf{x}_0, ..., \mathbf{x}_t\}$ and $\mathbf{y}_{1:t} \triangleq \{\mathbf{y}_1, ..., \mathbf{y}_t\}$, respectively. The measurements $\mathbf{y}_t$ are recorded by K sensors, and we use $\mathbf{y}_t^k$ to denote the subset of observations made by the k-th sensor. The main goal is to estimate online the expected value of the current state $\mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}[\mathbf{x}_t]$.

The parallel distributed particle filter algorithm is detailed below.

1. Initialization, $t = 0$
   - Initialize the particle filter of each sensor $k = 1, ..., K$ using the same random seed.
   - For each sensor $k = 1, ..., K$
     - For $i = 1, ..., N$ particles, sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$.
   - Set $t = 1$.
2. Quantization and encoding at the nodes.
   For the length of the vector $v = 1, ..., V$:
   (a) Quantization
   - For each sensor $k = 1, .., K$
     - For $i = 1, .., N$, sample $\tilde{\mathbf{x}}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)})$.
     - Calculate expected values of measurements $g_t^{(i)} = E(y_t^{(k)}|\mathbf{x}_t^{(i)})$.
     - Create a histogram of the $g_t^{(i)}$ values with h equal bins encompassing the range of values $[\min(\mathbf{g}_t), \max(\mathbf{g}_t)]$.
     - Use the histogram to form Huffman tree H_t^k and encode quantized measurements $\tilde{\mathbf{y}}_t^k$.

   (b) Local Estimation
   - Form a *local* estimate of the object's state using a standard particle filter acting only on the local measurements.

   (c) Set $t \leftarrow t + 1$.
3. Network Communication:
   - For each $k = \{1, ..., K\}$, send the vectorized measurements $(\widetilde{\mathbf{y}}_{t-V:t}^k)' = \{(\widetilde{\mathbf{y}}_{t-V}^k)', ..., (\widetilde{\mathbf{y}}_t^k)'\}$ to all other $K - 1$ sensors.
4. Global Estimate:
   - For $t' = t - V, ..., t$
     (a) For each active class A node k, create the Huffman tree $H_{t'}^k$ to reconstruct the quantized data $\widetilde{\mathbf{y}}_{t-V:t}^k$.
     (b) Using $\{\widetilde{\mathbf{y}}_{t-V:t}^k, k = 1, ..., K\}$ as the set of measurements obtained for time interval $t - V : t$, apply a standard particle filtering algorithm to generate the *global* state estimates.
5. Go to step 2.

5. SIMULATIONS

We conducted simulations to explore the performance of the PDPF algorithm relative to a centralized particle filter. The simulations used a sensor network of 16 class A nodes, placed in a uniform 4×4 grid in a 128×128 metre plane. There are 128 uniformly distributed class B sensors dispersed to take distance measurements. Each of these class B nodes is associated with exactly one Class A node.

The dynamic system of the object's movement uses a jump-state Markov model, described by an initial distribution $p(u_0, \theta_0, \mathbf{x}_0)$ and update equations

$$u_t \sim p(u_t|u_{t-1}), \tag{1}$$

$$\theta_t = \theta_{t-1} + c(u_t) + v_t, \tag{2}$$

$$x_t = x_{t-1} + m[\cos\theta_t, \sin\theta_t], \tag{3}$$

where $u_t \in \{0, 1, 2\}$ represents a discrete motion state of the object (continuing straight, making a 0.1 radian left turn or making a 0.1 radian right turn, respectively). $c(u_t)$ represents the angle of turn in radians. The angle of the motion is represented by θ_t, which has a zero-mean Gaussian innovation noise v_t of variance 0.001. The object's position is x_t and has a constant speed $m = 0.5$. The update probability matrix for the discrete state is

$$p(u_t|u_{t-1}) = \begin{pmatrix} 0.75 & 0.65 & 0.65 \\ 0.125 & 0.3 & 0.05 \\ 0.125 & 0.05 & 0.03 \end{pmatrix}$$

The observation equation for node v with position g_v is

$$r_t^v = \max(\|x_t - g_v\|(1 + s_t), 0), \tag{4}$$

where s_t is zero-mean Gaussian noise of variance $\sigma_s^2 = 0.02$. Note eight class B sensors are active at any given time.

The simulations were conducted over $S = 20$ different realizations of sensor field and object path. In each study, the algorithm was applied with $REP = 5$ different random seeds for initialization. We report results for the case of $N = 500$ particles and $Q = 8$ quantization levels (bins). More extensive experiments, as reported in [13], indicate that these values provide a reasonable compromise between tracking accuracy and compression.

5.1. Experimental Results

In order to evaluate the tracking accuracy of the proposed algorithm in these two situations, we use the mean square error (MSE). The MSE is defined as

$$MSE = \frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{REP}\sum_{r=1}^{REP}\|x_{t,s} - \hat{x}_{t,r,s}\|^2\right] \tag{5}$$

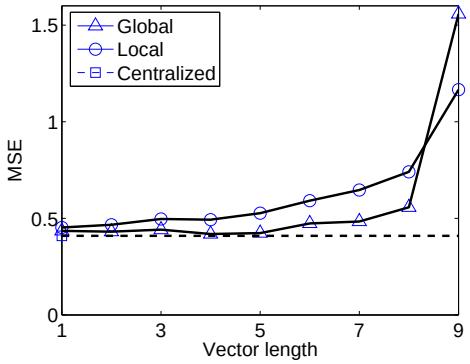

Fig. 2. Average mean square error for the global and local PDPF, and for the centralized particle filter for various vector lengths using 500 particles and 8 quantization bins.

where $x_{t,s}$ is the vector position of the object at time t and $\hat{x}_{t,r,s}$ represents the particle filter estimate.

The average MSE results of Figure 2 indicate that the accuracy of the PDPF global estimates for vector lengths up to 8 is very similar to that of the centralized particle filter and the distributed particle filter algorithm of [8] (i.e., vector length 1). This indicates that in the case where an estimate is not needed at every time step, vectorization can be used without incurring a substantial penalty in tracking accuracy. Of course, these results are specific to the reported tracking scenario, and the feasible vector length (in this case 8) reduces when the innovation or measurement noise increase. In the reported simulation, the active class B sensors are only changed when communication is exchanged. This means that measurements become more unreliable as the vector length increases, because the object is likely to be further away from the sensors. Figure 2 also shows the average MSE of the *local* estimates. In this simulation, these values are only slightly larger than the global estimates, because there is often a substantial number of distance measurements available to each class A node.

5.2. Communication and Computation

Here we compare the communication and computational requirements of the parallel distributed particle filter with those of a centralized scheme in which measurements are quantized to 16 bit values. We observe in Figure 3 that vectorization generates a substantial savings in the average number of bits transmitted. This is because communication between class A nodes occurs only once every period T. With a period of T, there are $b(T-1)$ packet overhead bits saved, where b is the number of bits per header.

In the centralized case, we assume the bulk of communication required in the network is from each individual class A node to the central fusion node. If there are m active class A node that transmit n quantized measurements bits,

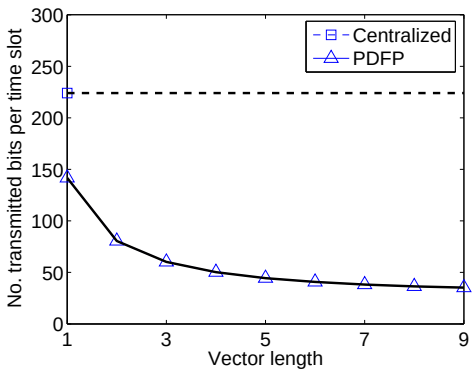

Fig. 3. Avg. number of bits sent in the network per time slot using various vector lengths for 500 particles and 8 quantization bins. The centralized scheme assumes 16-bit values are sent.

the communication cost is $O(m(n+b))$ per time step. In the distributed case with quantization and vectorization, the required number of transmitted bits is $O(m(n+b/T))$. As T grows, the number of transmitted bits tends to $O(mn)$.

The scheme of [9] shares at each time step the parameters of a Gaussian mixture approximation of the particle set. This requires $O(cm(n_g+b))$ bits, where c represents the number of Gaussians in the mixture, and n_g represents the number of bits required to encode the mixture parameters. Whether this approach involves a smaller communication overhead depends on whether state dimension is small compared to data dimension (per time step) and whether the Gaussian mixture approximation is reasonably efficient. In our simulation scenario, the dimensions are comparable (8 measurements, 4 state components), so transmitting compressed, encoded data is much more efficient. However, if the data were images from a camera, then the Gaussian approach would undoubtedly be preferable.

The PDPF algorithm is inherently more computationally expensive than the centralized approach because a particle filter must be maintained at multiple nodes. The maintenance of each of these particle filters also requires more computation, because there is the extra quantization and encoding procedure. The particle filter is known to computationally cost $O(N)$ if N particles are used. Blind particle propagation in the encoding also costs $O(N)$. Forming the histogram and creating the Huffman tree can be performed with a quicksort algorithm of complexity $O(N \log N)$. The dominant computational cost in the PDPF is not the quantization/encoding. In our implementation, particle propagation only consumes $\frac{1}{15}$ of the computation time of the standard particle filter component. The encoding (decoding) takes approximately $\frac{1}{4}$ of the time of the particle filter. The maintenance of the parallel particle filter at any class A node thus requires between 1.25 - 1.5 times the computation of a standard particle filter.

6. CONCLUSION

Centralized particle filtering in a sensor network suffers from a single point of failure and unbalanced, large communication requirements. Previously proposed distributed particle filters alleviate these problems but fail to consider the substantial overhead of packet header bits. In this paper, we have proposed a vectorization approach that allows multiple nodes to run parallel particle filters and share measurements extremely efficiently. Tracking accuracy is maintained and communication overhead dramatically reduced. Further work is needed in the development of distributed particle filtering architectures to address issues of unreliable measurements, data association, and sensor selection.

7. REFERENCES

[1] A.H. Jazwinski, *Stochastic processes and filtering theory*, Academic Press, New York, 1970.

[2] R. R. Brooks, P. Ramanathan, and A.M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proc. IEEE*, vol. 91, pp. 1163–1171, Aug. 2003.

[3] D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Trans. Automatic Control*, vol. 17, pp. 439–448, Aug. 1972.

[4] F. Martinerie, "Data fusion and tracking using HMMs in a distributed sensor network," *IEEE Trans. Aerospace and Electronic Systems*, vol. 33, pp. 11–28, Jan. 1997.

[5] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods*, Springer-Verlag, New York, 2001.

[6] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Sig. Processing*, vol. 50, pp. 174–188, Feb. 2002.

[7] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, vol. 19, pp. 61–72, Mar. 2002.

[8] M.J. Coates, "Distributed particle filtering for sensor networks," in *Proc. IEEE/ACM Int. Symp. IPSN*, Berkeley, CA, Apr. 2004.

[9] X. Sheng and Y-H. Hu, "Distributed particle filter with GMM approximation for multiple target localization and tracking in wireless sensor network," in *Proc. IEEE/ACM Int. Symp. IPSN*, Los Angeles, CA, Apr. 2005.

[10] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Proc. UAI*, Acapulco, Mexico, Aug. 2003.

[11] J. Shin, N. Lee, S. Thrun, and L. Guibas, "Lazy inference on object identities in wireless sensor networks," in *Proc. IEEE/ACM Int. Symp. IPSN*, Los Angeles, CA, Apr. 2005.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Mass., 2nd edition, 2001.

[13] G. Ing and M.J. Coates, "Vectorized distributed particle filtering," Tech. Rep., Dept. Electrical and Computer Engineering, McGill University, Apr. 2005.