INTELLIGENT SENSOR FUSION: A GRAPHICAL MODEL APPROACH

José M. F. Moura, Jin Lu, and Marius Kleiner

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
[moura,jinlu]@ece.cmu.edu

ABSTRACT

We study the fusion of data collected by multiple heterogeneous sensors that work cooperatively to achieve a common goal. This paper presents fast algorithms to fuse the sensor data. We map the problem into a graphical model and then develop a fast message-passing scheme to fuse the data. We simulate scenarios with 150 sensors and 200 targets that are successfully fused.

1. INTRODUCTION

Our work is relevant to many alternative scenarios and applications. To be speci£c, we focus on the tracking of multiple, possibly moving, targets by an ad-hoc network of autonomous, expendable sensors.

There are many issues related to the management of such ad-hoc network of sensors. We deal here with one, namely, with deriving a fast algorithm to fuse across the sensors the information extracted by each sensor from their local data. We do not consider related important issues that derive from bandwidth, computational, or power constraints, [1], that limit the operation of each individual sensor.

Soft decision: Heterogeneous sensors The sensors in the network may be of different physical types, possibly spanning several sensing modalities, e.g., acoustics, electromagnetic, or infrared; they can be point sensors or arrays of sensors; some may provide high resolution while others only coarse resolution; some sensors may be omnidirectional, while others may exhibit some level of directionality. This heterogeneity raises the issue of how to integrate the data from such diversity of modalities and resolutions. We fuse the information provided by the sensors not on the physical space of their individual measurements, but on the logical layer of their outputs. For example, to track targets with seismic sensors, we may move from the pressure or vibration signals at the front-end to the intermediate (still physical space) of features (spectral lines), and then to the logical space in the back-end of "soft decisions"—the likelihood that each target is at a given location, given the physical measurements. These likelihoods are conditional probabilities.

This work was partially supported by the DARPA ISP Program through the Army Research Of£ce grant #DAAD19-02-1-0180 and by NSF through grant # ECS-0225449.

Probabilistic inference on graphical models Most likely, individual sensors cannot make a reliable determination regarding the position of the targets on their own, but fusion of the appropriate sensors may provide reliable information about their locations. Fusing the soft decisions of the sensors is challenging because the sensors are local, survey areas that are only partially overlapping, have different resolutions and sensing ranges, and may exhibit intricate probabilistic dependencies. To achieve global space awareness, we need to integrate this disparate soft information into a coherent global framework. Graphical models are particularly good at capturing these assorted soft decisions with diverse interdependencies, [2, 3]. Our goal is to go from the set of individual sensors' soft decisions—say the likelihood of M targets being detected, as computed by a given sensor—to the likelihood of a given target being detected, based on all sensors relevant to that target. We capture this integration of local partial views into a probabilistic inference problem on graphical models. These algorithms have received much attention in turbo and low density parity check (LDPC) decoding, [4, 5, 6].

2. SENSOR FUSION: INFERENCE IN FACTOR GRAPHS

To illustrate the approach we develop a simple example of three targets $\{T_i,\ i = 1, 2, 3\}$ being sensed by three sensors $\{S_j,\ j = 1, 2, 3\}$. Sensor S_1 measures targets T_1 and T_2; sensor S_2 detects targets T_2 and T_3; and, £nally, sensor S_3 monitors targets T_1 and T_3. The above relationships between sensors and targets are shown in £gure 1. Rather than outputting only the most probable location of each target, the sensors provide the probability of the targets being at each of the different possible locations. This probabilistic information, soft information, is in the form of a conditional probability, in our example, $p(T_1, T_2 \mid S_1)$, $p(T_2, T_3 \mid S_2)$ and $p(T_1, T_3 \mid S_3)$. The function $p(T_i, T_j \mid S_k)$ stands for the

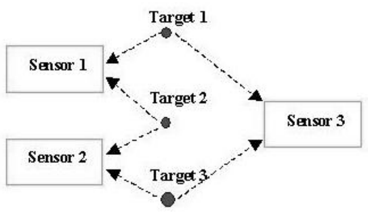

Fig. 1. Illustrative problem: 3 sensors and 3 targets

conditional probability that targets i and j are at positions T_i and T_j given the data collected by sensor S_k. The fusion goal is to derive for each target the conditional probability of its position based on the data collected by all sensors, i.e., $p(T_i \mid S_1, S_2, S_3)$. These $p(T_i \mid S_1, S_2, S_3)$, $i = 1, 2, 3$ are the marginal probability functions of the joint probability function $p(T_1, T_2, T_3 \mid S_1, S_2, S_3)$, e.g.,

$$p(T_1 \mid S_1, S_2, S_3) = \sum_{T_2, T_3} p(T_1, T_2, T_3 \mid S_1, S_2, S_3). \quad (1)$$

The problem with the marginalization in (1) is that it is of order $O\left(M^2 L^M\right)$, where M is the number of targets and L is the number of resolution beams (assuming all sensors have similar resolution).

We avoid the direct computation of the joint probability $p(T_1, T_2, T_3 \mid S_1, S_2, S_3)$ and then its marginalization by computing the marginals through a graphical model called *factor graph*. A factor graph is a bipartite graph containing two types of nodes: *variable nodes* and *function nodes*. A variable node x is connected to a function node f by an edge if and only if x is an argument of f. This model leads to an iterative algorithm to compute function marginals. This iterative algorithm, the *sum-product algorithm*, computes the marginals by passing messages on the factor graph.

To map the problem into the factor graph, we £nd £rst a multiplicative decomposition of the joint probability function, usually referred to as the global function to distinguish it from the functions f that are the local functions. In this problem, the global function is $p(T_1, T_2, T_3 \mid S_1, S_2, S_3)$. Assuming that the targets $\{T_i,\ i = 1, 2, 3\}$ are independent of each other and that the noises in the sensors $\{S_j,\ j = 1, 2, 3\}$ are also independent, the global function factors into a product of six local functions, as shown in equation (2).

$$p(T_1, T_2, T_3 | S_1, S_2, S_3) = \quad (2)$$
$$\underbrace{p(T_1, T_2|S_1)}_{f_1}\underbrace{p(T_2, T_3|S_2)}_{f_2}\underbrace{p(T_1, T_3|S_3)}_{f_3}\underbrace{\frac{1}{p(T_1)}}_{g_1}\underbrace{\frac{1}{p(T_2)}}_{g_2}\underbrace{\frac{1}{p(T_3)}}_{g_3}$$

Since we have three variables $\{T_i,\ i = 1, 2, 3\}$ ($\{S_j,\ j = 1, 2, 3\}$ are just labels, not considered to be variables) and six local functions $\{f_i,\ i = 1, 2, 3\}$ and $\{g_j,\ j = 1, 2, 3\}$, the factor graph for this problem is composed of 3 variable nodes and 6 function nodes, as illustrated in £gure 2.

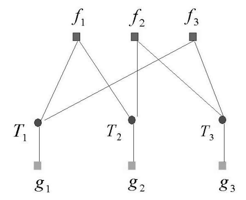

Fig. 2. Factor graph for problem in £gure 1.

3. FAST SUM-PRODUCT ALGORITHM

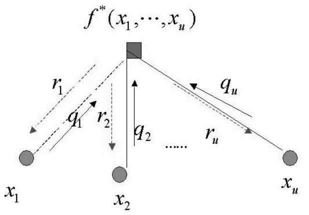

Fig. 3. Message updating around function node

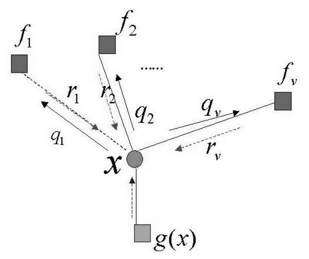

Fig. 4. Message updating around variable node

The sum-product algorithm is computationally demanding when the degree of the function nodes is high. For a function node $f^*(x_1, x_2, \ldots, x_u)$ connected by u variable nodes $\{x_i,\ i = 1, 2, \ldots, u\}$ in £gure 3, the task of the message updating is to compute all the outgoing messages $\{r_i(x_i),\ i = 1, 2, \ldots, u\}$ with respect to all variables x_i, $i = 1, 2, \ldots, u$, i.e., $r_1(x_1), \ldots, r_u(x_u)$, where

$$r_i(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_u} \{f^* \prod_{j=1,2,\ldots,u \text{AND} j \neq i} q_j(x_j)\} \quad (3)$$

In equation (3), the symbol $q_j(x_j)$ denotes the incoming message transmitted from the variable node x_j to the function node f^*. We assume each variable x_i takes L different values so the u-argument function $f^*(x_1, x_2, \ldots, x_u)$ is represented by a u-dimensional array with each dimension having a resolution of L points. To multiply two incoming messages $q_1(x_1)$ and $q_2(x_2)$ to generate a $L \times L$ array $\pi_{12}(x_1, x_2)$, we need to multiply every element of the vector $q_1(x_1)$ with every element of the vector $q_2(x_2)$.

Hence, L^2 point-wise multiplications are required. In the subsequent step, to multiply $\pi_{12}(x_1, x_2)$ with $q_3(x_3)$, we need again to multiply every element of $\pi_{12}(x_1, x_2)$ with every element of $q_3(x_3)$. Since $\pi_{12}(x_1, x_2)$ has L^2 elements and $q_3(x_3)$ has L elements, $L^2 \times L = L^3$ point-wise multiplications are needed. Extending the above reasoning, we £nd that $\sum_{k=2}^{u} L^k$ multiplications are needed to compute $f^*(x_1, x_2, \ldots, x_u)\prod_{j=1,2,\ldots,u \text{ AND } j\neq i} q_j(x_j)$. After multiplying the required incoming messages with the local function, we still need to compute the function marginal. Initially, to sum out x_u, we need to add all possible values of x_u for all different combinations of $x_1, x_2, \ldots, x_{u-1}$. Since there are L^{u-1} combinations for $x_1, x_2, \ldots, x_{u-1}$ and for each combination $L-1$ additions are required, we need to perform $L^{u-1} \times (L-1)$ sums. This is repeated with the other variables, till a function of a single argument is derived. This leads to $\sum_{k=2}^{u} L^{k-1}(L-1) = L^u - L$ sum. This is only for a single outgoing message; the cost to compute all u outgoing messages is $u\sum_{k=2}^{u} L^k$ multiplications and $u(L^u - L)$ additions. In brief, the computational cost is $O(uL^u)$ when $u \gg 1$.

We now present a *divide-and-conquer* algorithm to reduce this computational cost. We assume the simple case where the function f^* in £gure 3 has only 4 arguments: x_1, x_2, x_3 and x_4. Instead of computing directly $\{r_i(x_i)\ i = 1, 2, \ldots, 4\}$ from equation (3), we do the following: split all four variables into two groups, $\{x_1, x_2\}$ and $\{x_3, x_4\}$, and compute $\pi_{12} = q_1(x_1)q_2(x_2)$ and $\pi_{34} = q_3(x_3)q_4(x_4)$. Next compute $\{r_i(x_i)\ i = 1, 2, \ldots, 4\}$ using the revised message-updating equations (4-7):

$$r_1(x_1) = \sum_{x_2} q_2(x_2)\{\sum_{x_3}\sum_{x_4}(\pi_{34} f^*(x_1, x_2, x_3, x_4))\} \quad (4)$$

$$r_2(x_2) = \sum_{x_1} q_1(x_1)\{\sum_{x_3}\sum_{x_4}(\pi_{34} f^*(x_1, x_2, x_3, x_4))\} \quad (5)$$

$$r_3(x_3) = \sum_{x_4} q_4(x_4)\{\sum_{x_1}\sum_{x_2}(\pi_{12} f^*(x_1, x_2, x_3, x_4))\} \quad (6)$$

$$r_4(x_4) = \sum_{x_3} q_3(x_3)\{\sum_{x_1}\sum_{x_2}(\pi_{12} f^*(x_1, x_2, x_3, x_4))\} \quad (7)$$

Though $\sum_{x_3}\sum_{x_4}(\pi_{34} f^*(x_1, x_2, x_3, x_4))$ appears twice in (4) and (5), we actually only need to compute it once. Similarly with the term $\sum_{x_1}\sum_{x_2}(\pi_{12} f^*(x_1, x_2, x_3, x_4))$. Doing so saves on the number of actual ¤oating point operations (Flops) needed. The exact number of multiplications needed for this divide-and-conquer algorithm is $2L^4 + 6L^2$ and the number of additions is $2L^4 + 2L^2 - 4L$ where L is the sensors resolution. As the standard sum-product algorithm requires $4L^4 + 4L^3 + 4L^2$ multiplications and $4L^4 - 4L$ additions, the divide-and-conquer strategy decreases the computational cost by a factor of two with no loss in accuracy since we only rearrange the order of the multiplications and sums. These arguments are easily generalized to functions with an arbitrary number of variables. Initially, all variables are divided into two subsets as evenly as possible, then each of the two subsets are further split in half as uniformly as possible. This argument is repeated till the number of variables contained in each subset is one or two. Next, the sums and multiplications are rearranged to let the message-updating equations share the same terms as much as possible. To compute the computational savings is rather long and tedious and will not be detailed here due to lack of space. The conclusion is that the divide and conquer algorithm is faster than the standard sum-product algorithm by a factor of $\frac{u}{2}$ where u is the degree of the function node.

An alternative algorithm to reduce the computational cost is *multiply-and-divide*. First, we compute the products of all the incoming messages with the local function using equation (8):

$$\pi = f^*(x_1, \ldots, x_u)\prod_{j=1}^{u} q_j(x_j) \quad (8)$$

After π in equation (8) is known, we update each outgoing message simply using the next equation

$$r_i(x_i) = \frac{1}{q_i(x_i)}\sum_{x_1}\cdots\sum_{x_{i-1}}\sum_{x_{i+1}}\cdots\sum_{x_u}\pi \quad i = 1, 2, \ldots, u$$

Since the same term π is used u times when updating outgoing messages $r_i(x_i)\ i = 1, 2, \ldots, u$, the number of multiplications needed is $O(2L^u)$ whereas the number of sums needed is still the same as with the standard sum-product algorithm, which is $O(uL^u)$ when $u \gg 1$. In summary, of the three message-updating algorithms discussed the divide-and-conquer strategy is the most computationally ef£cient.

The analysis above studied the message updating around the function nodes. We next consider the message updating around the variable nodes, as shown in £gure 4. We can show that the multiply-and-divide strategy is now the most ef£cient. For example, if we let v denote the degree of the variable nodes and L the resolution of the variables the multiply-and-divide is $O(2vL)$ whereas the standard sum-product algorithm has a complexity $O(v^2L)$.

4. SIMULATION RESULTS

We apply the divide-and-conquer strategy to update messages around function nodes and the multiply-and-divide strategy when updating messages around variable nodes.
Experiment I (Convergence Study) Three sensors and three targets are considered in this experiment. Their relationships are shown in £gure 1. The sensing resolution is set to be $L = 100$. The prior probability functions $\{g_j,\ j = 1, 2, 3\}$ are uniform distributions while the soft information acquired by the sensors, i.e., $\{f_i,\ i = 1, 2, 3\}$, is taken to be a mixture of two Gaussians with different means and variances. The fast sum-product algorithm terminates when the mean square difference between the outputs of two successive iterations is less than a preset threshold, or when the running epochs exceed the maximum allowed number of iterations.

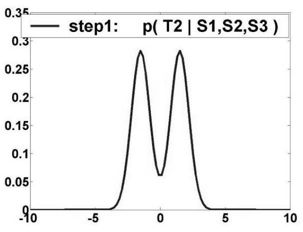

Fig. 5. (a)

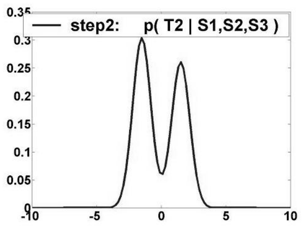

Fig. 5. (b)

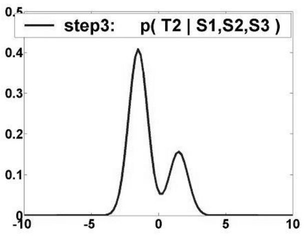

Fig. 5. (c)

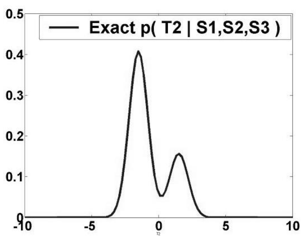

Fig. 5. (d)

Fig. 5. Convergence study of the fusion algorithm

Figure 5(a) shows the temporary result corresponding to the target T_2 after the £rst iteration, whereas £gures 5(b) and 5(c) show the temporary results for the target T_2 after the second and the third iteration respectively. Figure 5(d) is the exact result of for the target T_2 when computing the function marginal directly by marginalization of the joint. From £gures 5(a)-5(c), we conclude that the sum-product algorithm converges fast and that the result of the third iteration matches the exact result remarkably well.

Experiment II (Small Sensor Network with High Sensing Resolution) We now consider a network containing 15 sensors surveying 20 targets. The relationship between sensors and targets, i.e., which targets are being sensed by a speci£c sensor, is randomly generated. Each sensor can detect 4 targets while each target is sensed by 3 different sensors. We choose the sensing resolution $L = 50$. Again, the prior probability functions are assumed to be uniform. The soft information provided by the sensors are four-dimensional Gaussians with randomly generated mean vector and covariance matrix. We report the result for target T_4 after 3 iterations. The output $p(T_4 \,|\, S_1, S_9, S_{14})$ is shown as the solid curve in £gure 6. We also include the probability distribution functions $p(T_4 \,|\, S_1)$, $p(T_4 \,|\, S_9)$ and $p(T_4 \,|\, S_{14})$ depicted by the dashed curves. The plot shows that the sensor network has been successfully integrated. We notice that the fusion result $p(T_4 \,|\, S_1, S_9, S_{14})$ has a smaller variance.

Experiment III (Large-scale Sensor Network with Low Sensing Resolution) We have now 150 sensors and 200 targets. Each sensor can detect 4 targets, while every target is detected by 3 sensors. All the other settings are the same as in experiment II except that the sensing resolution is set to $L = 10$. The results for the target T_3 are provided in

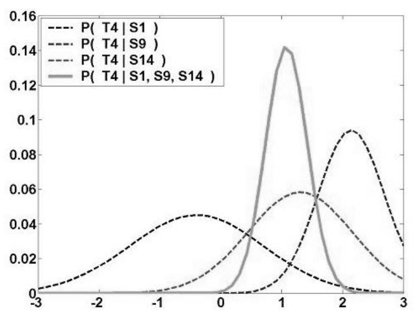

Fig. 6. One of 20 fusion results for experiment II

£gure 7 (solid curve) after 3 iterations. Again, the probability distribution functions $p(T_3 \,|\, S_{15})$, $p(T_3 \,|\, S_{100})$, and $p(T_3 \,|\, S_{123})$ are presented by the dashed curves. We £nd that even though the sensing resolution is low, the resulting output still fuses the soft information provided by the sensor network.

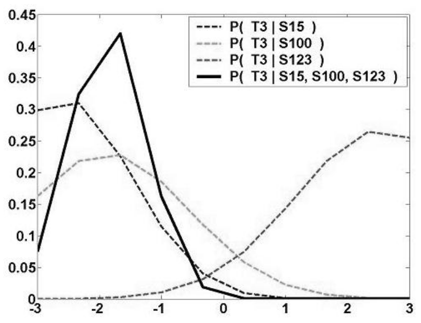

Fig. 7. One of 200 fusion results for experiment III

5. REFERENCES

[1] J. M. F. Moura, R. Negi, and M. Pueschel. *Distributed sensing and processing: a graphical model approach.* DARPA ACMP Integrated Sensing and Processing Workshop, Sep. 18/ 2002, Annapolis, MD.

[2] B. J. Frey. *Graphical models for machine learning and digital comm.* MIT Press, Cambridge, MA, 1998.

[3] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. *Factor graphs and the sum-product algorithm.* IEEE Trans. Inform. Theory, 47, pp. 498-519, Feb. 2001.

[4] C. Berrou and A. Glavieux. *Near Optimum Error Correcting Coding and Decoding: Turbo codes.* IEEE Trans. on Comm., 44(10), pp. 1261-1271, Oct. 1996.

[5] R. G. Gallager. *Low-Density Parity Check Codes.* No. 21 in Res. Monograph Series. Cambridge, MA: MIT Press, 1963.

[6] D. J. C. Mackay. *Good Error-Correcting Codes Based on Very Sparse Matrices.* IEEE Trans. on Inform. Theory, 45(2), pp. 399-431, March 1999.