Data fusion without data fusion: localization and tracking without sharing sensitive information

Matthew Roughan [1] and Jon Arnold [2]

[1] *School of Mathematical Science*
University of Adelaide, SA 5005, Australia
matthew.roughan@adelaide.edu.au

[2] *Defence Science and Technology Organization, Australia,*
jon.arnold@dsto.defence.gov.au

Abstract

It is now well known that data-fusion can improve detection and localisation of targets. However, traditional data fusion requires the sharing of detailed data from multiple sources. In some cases, the various sources may not be willing to share such detailed information. For instance, current allies may be willing to share some level of information, but only if they can do so without revealing their secrets. We show here that, at least for localisation and tracking of targets, data-fusion can be performed without the need to actually combine the data in question, so that no party learns the information of another. Such an approach would allow co-operation between parties that share mutual interests, and yet do not completely trust each other. The particular application on which we concentrate is localisation and tracking of a target using multiple sensors (radars or sonars, or other devices). We show that multiple sensors can be used to refine a target's position estimate, or even obtain a position estimate where no single sensor has enough information to do so (e.g., each has only range information), without sharing the details of the sensor, such as its position, or accuracy of its estimates.

1. INTRODUCTION

It is now a standard data-fusion problem to use multiple sensors to improve the localisation and subsequent tracking of targets. However, there may be cases where such co-operation is limited by the nature of the parties who wish to co-operate. For instance, consider several parties who wish to be able to detect illegal fishing, drug smuggling, or terrorist activities. They may use military radar in some cases to aid in such detection. They certainly would not want details of their military radar to be unnecessarily revealed, even to current allies, given that their relationship may change in the future. Hence we have a problem: how can we do data fusion, without fusing the data?

In fact there are many similar problems: for instance it has been recently noted that medical information may be held in multiple databases, stored by different parties who are unable to share data for commercial or legislative reasons (*e.g.*, privacy legislation). There are now multiple techniques in the Privacy-Preserving Data Mining (PPDM) literature for performing data mining on such distributed databases [1]–[4], [8]–[10]. We can apply such techniques to the problem at hand.

We shall consider two problems here. First we consider a problem where each party has estimates of a target's position. They then wish to combine this information to provide a better estimate of the target's location without revealing information about their sensors. The approach we use is a *secure distributed summation* (SDS). The accuracy of estimates is as good as if we were not concerned with security and computed the estimates directly, but each parties' input information is not revealed to the other parties. SDSs can also be used for weighted averages where different sensors have different accuracy, and we can do so without revealing the accuracy of the sensors. We extend this work to consider tracking in this environment, and although we can use a standard tracking algorithm (the Kalman filter), we are required to adapt the SDS to allow for the fact that we perform it multiple times over the track.

The second problem we consider is one where any one sensor doesn't have enough information (in itself) to localise a target. The example we consider here is where each sensor provides range measurements. In itself, such information is inadequate to localise a target, but in combination with data from other sensors, one can provide good position estimates. This type of approach appears particularly relevant in the context of ad-hoc networks where we may wish to localise some resource on the network from purely passive measurements of signal strength at a number of points [5], [7], and each node in the network could be controlled by a separate party.

The contributions of this paper are the application of PPDM to a new problem which has not been seen before (to our knowledge); and a new PPDM algorithm for computing the location of a target from range estimates without sharing detailed information about the positions of sensors. We consider tracking in both cases, but the tracking problems are rather standard once the PPDM algorithms are designed, with the

exception that some care must be taken in the design of PPDM algorithms when they are to be performed iteratively (such as in tracking) and we show how to do so, including showing how to compute the covariance of the measurements using a PPDM approach.

2. Problems and assumptions

A. Problem 1

We first consider a simple problem where N parties each use (potentially multiple) sensors to create an estimate of the position of a single target. We denote the position of the target by (x, y) (relative to some arbitrarily chosen, but agreed point), and the estimate from party i by $(\hat{x}_i, \hat{y}_i)$. We assume that the position estimate has negligible bias and that the errors in position have covariance matrix S_i. Given constant variance $S_i = S$, we might improve our estimate of the position of the target by taking

$$\hat{x} = \frac{1}{N}\sum_{i=1}^{N}\hat{x}_i, \quad (1)$$

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N}\hat{y}_i. \quad (2)$$

The natural approach to computing the sum would be for each party in the measurement to pool values and then compute the sum. This approach reveals at least some values to other parties. An alternative would be to use a trusted third party to pool the results, and hence keep them secret. However trusted third parties are not easy to find.

We typically wish to extend the problem to the case where S_i varies between sensors, at which point we have to allow weighted means to be computed. However, the parties in the computation may not wish to reveal their value of S_i as this reveals important information about their sensor. Another extension of the problem occurs when we have multiple measurements over time, and we wish to track the target.

B. Problem 2

In the second problem we allow the N parties to make only range estimates. The combination of two such estimates is enough to localise the target to two possible points, and three or more such measurements are capable of deducing the position uniquely (with some rare exceptions). It is noteworthy, however, that when the measurements contain errors, the measurements may be inconsistent, resulting in a problem in estimating the target's position precisely. We denote range estimates from party i by D_i (which is an unbiased estimate of the true range d_i), and the position of the sensor of party i by (X_i, Y_i).

C. Assumptions

In any PPDM we must define the security model under consideration. We will assume here the commonly used "honest-but-curious" model. That is, we assume that the co-operating parties are honest in the sense that they follow the algorithms correctly, but they are curious and they will perform additional operations in order to attempt to discover more information than intended. The honest-but-curious assumption has been widely used, and appears applicable here. Sensor operators will benefit from participating honestly in such a scheme without revealing their private information, and there is no downside in participating honestly. Dishonest partners in computation (partners who do not follow the algorithm) will reduce their own benefits, without any obvious gain.

It is noteworthy that while we assume that participants follow the algorithm correctly, we do allow collusion. Multiple partners are allowed to collude to attempt to learn more information than they otherwise could. The protocols we present can be made resistant to such collusion in the presence of a majority of non-colluding participants. Additionally, there is now a substantial literature on PPDM (*e.g.* see [1]–[4], [8]–[10] and the references therein), and this literature considers many variations on the type of assumptions considered here. It is therefore likely that the assumption of honest-but-curious participants can be substantially weakened, though we leave this as a topic for future research.

3. The Solution: problem 1

The problem amounts to a distributed summation (a summation over a distributed set of data points), for which approaches are now well understood (see [4], [6] and the references therein). There are at least three approaches to the problem with varying degrees of efficiency and robustness to collusion. We describe the simplest here for the purpose of exposition. The problem is a special case of the problem of computing $V = \sum_{j=1}^{N} v_j$, where the individual values v_j are kept by party j, and considered to be secret. We can perform the distributed summation as follows. Assume the value V is known to lie in the interval $[0, n]$, where n may be large. Start from a particular party (we will denote this party 0), and list the other parties in some order with labels $1, 2, \ldots, N-1$, then perform:

party 0: randomly generate $R \sim U(0, n)$
party 0: compute $s_1 = v_1 + R \mod n$
party 0: pass s_1 to party 1
for i=1 to N-1
  party i: compute $s_i = s_{i-1} + v_i \mod n$
  party i: pass s_i to party $i+1 \mod N$
endfor
party 0: compute $V = s_{N-1} - R \mod n$
party 0: share V with other parties

Each party $i = 1, \ldots, N-1$ has only the information v_i and s_{i-1}, which can be written in full as

$$s_i = R + \sum_{j=1}^{i} v_j \mod n. \quad (3)$$

Since this value is uniformly distributed across the interval $[0, n]$, party i learns nothing about the other values v_j, $j \neq i$. At the last step, party 1 has s_N, and when it subtracts R away it gets the V. Note that where we wish to allow computation of quantities that may be negative, the condition that $V \in [0, n]$ can be easily replaced by $V \in [-n/2, n/2]$. The algorithm

above requires only that we adjust the range of the initial sum (*i.e.* in the second step party 1 takes $s_1 = n/2 + v_1 + R \mod n$), and that in the last step party 1 reverses this addition (*i.e.* $v_N = s_N - n/2 - R \mod n$).

Given V any party can compute $V - v_i = \sum_{j \neq i} v_i$, and so this approach only works for $N > 2$, and in reality, where one could make meaningful guesses about some values, it is only really secure for reasonable values of N, and this is the case we consider here. Other approaches can be applied to the case $N = 2$, *e.g.*, Yao's two-party protocol [9], [10].

This incredibly simple process can, unfortunately, be corrupted if parties collude. If party $l-1$ and party $l+1$ share information, they can compute v_l by taking $s_l - s_{l-1}$ (s_l is received by party $l+1$, and s_{l-1} is sent by party $l-1$). There are various approaches to avoid this issue, as well as dealing with potentially unreliable partners in the summation, for instance see [6] for more discussion.

Once we compute the sum, it is a simple matter to compute the average of the location estimates, and distribute this value to all parties.

Although the above approach hides some information — the individual position estimates — the final result is that each sensor learns a positional estimate, and so hiding the individual data seems to make little sense. However, it is simple to adapt this technique to the case where the sensors in question have different characteristics, *e.g.*, accuracy. It makes a lot more sense for a sensor operator to wish to hide their sensor's characteristics from other operators. Consider the simple case where each sensor's estimate has covariance $S_i = \sigma_i^2 I$ where I is the identity matrix. We then compute a weighted mean

$$\hat{x} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \hat{x}_i, \quad (4)$$

$$\hat{y} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \hat{y}_i, \quad (5)$$

where the weights $w_i = 1/\sigma_i^2$. If an operator wants to conceal the characteristics of his sensor, they would wish to keep the weights w_i secret. This is easily accomplished by performing two SDSs (for each co-ordinate), one over the weighted position, and the other over the weights themselves.

A. Tracking

Most sensor operators wish to go beyond simple localisation of targets to track them. There are two approaches we could adopt here: (i) pre-fusion tracking, and (ii) post-fusion tracking. In approach (i) each operator tracks the targets, and them uses SDSs to combine the track information (including inferred velocities). In approach (ii) we combine location estimates, and then track the target.

Let us perform a simple comparison of the two approaches. We do so in the context of a simple Kalman filter, *i.e.*, we use a linear model for the target and measurements

$$s_{t+1} = Fs_t + w_t, \quad (6)$$

$$z_t = Hs_t + v_t, \quad (7)$$

where the state $s_n^T = [x, y, \dot{x}, \dot{y}]$ represents the target's position (x, y) and velocity $(\dot{x}, \dot{y})$, and

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad (8)$$

and the process noise w_t, and measurement noise v_t have covariance matrices Q and R respectively. The standard Kalman filter for obtaining and estimating the state $\hat{s}_t$ at time t is given by

$$\hat{s}_{t+1}^- = F\hat{s}_t, \quad (9)$$

$$P_{t+1}^- = FP_tF^T + Q, \quad (10)$$

$$K_{t+1} = P_{t+1}^- H^T (HP_{t+1}^- H^T + R)^{-1}, \quad (11)$$

$$\hat{s}_{t+1} = \hat{s}_{t+1}^- + K_{t+1}(z_{t+1} - H\hat{s}_{t+1}^-), \quad (12)$$

$$P_{t+1} = (I - K_{t+1}H)P_{t+1}^-. \quad (13)$$

Note that we explicitly compute the covariance of the estimates, P_t, at each time step. The covariance of each track in the pre-tracking approach will be given by P_t, and so the final covariance (after combining N such estimates) would be P_t/N. The covariance of the post-track approach can be derived by noting that the measurements would have noise with covariance matrix reduced to R/N, and by using this term in place of R in (11) above.

Figure 1 shows $P_t(1,1)$ for a simple case with $Q = qI$ for $q = 0.001$, $R = rI$ for $r = 0.01$ and $T = 0.1$. The figure shows that the pre-tracking approach outperforms the post-track approach, which is intuitively obvious given we are averaging the process and measurement noise in the pre-tracking approach, but only the measurement noise in the post-track approach.

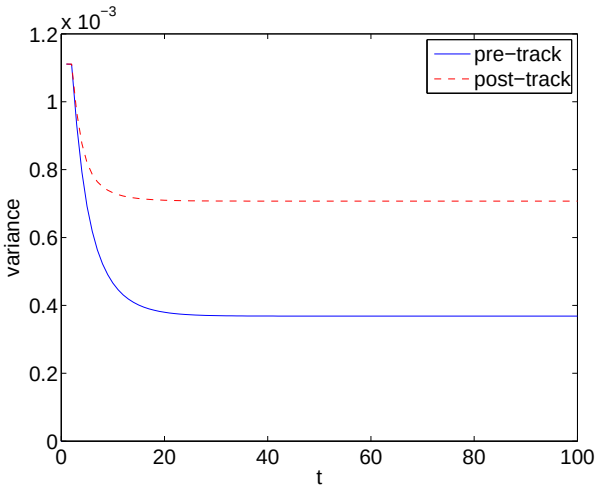


Fig. 1: An example variance estimate for the x co-ordinate for $q = 0.001$, $r = 0.01$ and $T = 0.1$.

The work above follows by now standard and well known tracking techniques. There are many standard extensions of such, for instance, it is common to use extended Kalman filters to deal with non-linearities in target motion, or measurements.

The above example is included for illustrative purposes, and contains nothing new. However, there is one new issue that arises in consideration of the privacy-preserving nature of our approach. One must take additional care with PPDM techniques when they are performed multiple times (as when tracking). For instance, if the first party in the above computation used a different random variable R in each SDS, then the second party could average over the series of measurements, and hence form an estimate of the characteristics of the first sensor's measurements. Hence, in performing tracking, a single random number R needs to be used in successive SDSs. On the other hand, given knowledge of a target's position, one can infer (over time) the value of R, from predictions of target position $\hat{s}^-_{t+1}$. Once R is known to some degree of accuracy, the security of the method is lost.

We can counter these particular attacks on the privacy of the algorithm by using the same types of techniques used to prevent collusion in SDSs. Collusion between party $l-1$ and $l+1$ allows them to compute v_l by taking $s_l - s_{l-1}$. A simple fix to this problem is provided in [4]: each party i randomly partitions v_i into M shares v_{im} such that

$$v_i = \sum_{m=1}^{M} v_{im}. \tag{14}$$

Secure summation is then performed M times to calculate the sum for each share individually, and the results added. However, the summation order is permuted for each share so that no party has the same neighbour twice. To compute v_i, the neighbours of i from every iteration would have to collude. With M shares, $2M$ colluding parties are therefore required to violate security.

Randomly partitioning the measurements into shares before averaging also avoids the issues involved with performing tracking on a target. It avoids the problem that the final estimate of position and each individual's measurement of position are directly related, and so we can use a different randomisation each time the SDSs are performed, hence removing the possibility of exploiting correlations over time to infer more than we wish to reveal.

4. SOLUTION: PROBLEM 2

The previous problem is not particularly challenging, as we can easily see how to apply SDSs to the problem. The second problem involves computing a location estimate for a target using range estimates from multiple parties who do not wish to reveal the location of their sensor, or its characteristics. This problem requires development of a new PPDM algorithm.

First, let us consider how we would solve this problem where we can share data. We will write out the solution to the problem in detail, in order to see how we can modify the approach to allow for the case where we don't wish to share data. The measurement data may contain errors, and so we use Minimum Mean Squared Error (MMSE) estimation of the location of the target. Define

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i, \tag{15}$$

$$\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i, \tag{16}$$

$$\sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N} (X_i - \bar{X})^2, \tag{17}$$

$$\sigma_Y^2 = \frac{1}{N}\sum_{i=1}^{N} (Y_i - \bar{Y})^2, \tag{18}$$

$$c_{XY} = \frac{1}{N}\sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y}), \tag{19}$$

$$\bar{V} = \frac{1}{N}\sum_{i=1}^{N} D_i^2 - X_i^2 - Y_i^2, \tag{20}$$

where D_i is the distance to the target as measured by the ith sensor, and we also define the column vectors $\mathbf{x} = (X_1, \ldots, X_N)^T$, $\mathbf{y} = (Y_1, \ldots, Y_N)^T$, and $\mathbf{1} = (1, 1, \ldots, 1)^T$. Then the problem of estimating the location of the target can be thought of as the problem of solving the N equations

$$D_i = \sqrt{(X_i - \hat{x})^2 + (Y_i - \hat{y})^2}, \quad i = 1, \ldots, N. \tag{21}$$

If we square the equations, and rearrange we get

$$D_i^2 - X_i^2 - Y_i^2 = -2(X_i\hat{x} + Y_i\hat{y}) + \hat{x}^2 + \hat{y}^2. \tag{22}$$

If we then sum the equations over i, and divide by N we obtain the equation

$$\bar{V} = -2\hat{x}\bar{X} - 2\hat{y}\bar{Y} + \hat{x}^2 + \hat{y}^2. \tag{23}$$

Subtracting (22) from (23) for each i we obtain the equations

$$v_i = 2(X_i - \bar{X})\hat{x} + 2(Y_i - \bar{Y})\hat{y}, \tag{24}$$

where $v_i = \bar{V} + X_i^2 + Y_i^2 - D_i^2$. We can rewrite the equations (24) as

$$\mathbf{v} = A\beta, \tag{25}$$

where $\mathbf{v}$ and $A = 2[\mathbf{x} - \mathbf{1}\bar{X}, \mathbf{y} - \mathbf{1}\bar{Y}]$ are known, and we wish to calculate $\beta = [\hat{x}, \hat{y}]^T$. However, note that there may be errors in $\mathbf{v}$, and so we seek a MMSE solution by pre-multiplying the equation by the Moore-Penrose pseudo-inverse, $A^+ = (A^TA)^{-1}A^T$ so that

$$\beta = \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = (A^TA)^{-1}A^T\mathbf{v}. \tag{26}$$

In the case where data can be shared, this is a standard solution to the problem of localising a node.

Now, consider the problem where we do not wish to share data. In particular, we do not wish to reveal (X_i, Y_i, D_i). It should by now be clear that we can use SDS to compute $\bar{X}$, $\bar{Y}$, σ_X^2, σ_Y^2, c_{XY}, and $\bar{V}$. Note also that

$$A^TA = 4N \begin{pmatrix} \sigma_X^2 & c_{XY} \\ c_{XY} & \sigma_Y^2 \end{pmatrix} \tag{27}$$

and hence we can compute

$$(A^T A)^{-1} = \frac{1}{4N(\sigma_X^2\sigma_Y^2 - c_{XY}^2)} \begin{pmatrix} \sigma_Y^2 & -c_{XY} \\ -c_{XY} & \sigma_X^2 \end{pmatrix}. \quad (28)$$

Hence, once the secure summation has been used to compute (15-20), all parties know $(A^T A)^{-1}$, however the rows of A are partitioned across all parties, and so we cannot directly compute $(A^T A)^{-1} A^T$. Instead, consider the computation of

$$A^T \mathbf{v} = 2 \begin{pmatrix} \sum_{i=1}^N v_i(X_i - \bar{X}) \\ \sum_{i=1}^N v_i(Y_i - \bar{Y}) \end{pmatrix}. \quad (29)$$

Both terms of $A^T \mathbf{v}$ may be computed once again using a SDS. At this point, all parties know $(A^T A)^{-1}$ and $A^T \mathbf{v}$, and it is therefore a trivial computation to determine β from (26).

As before, the solution has been obtained without sharing the individual data (such as sensor location) that the parties wish to remain secret. Note that the computational complexity of the approach is not much more than the computational complexity of the normal calculation, and the communications costs are still only $O(N)$, though they may be larger than those for the standard computation by a constant factor. However, note that if the operation is to be performed multiple times, such as in tracking an object, the quantities (15-19) need only be computed once at the start of the co-operative effort (for stationary sensors), and the only part that need be computed at each subsequent time interval is (29). This only requires two SDSs, which take only twice the communications of distributing the D_i, and so the communications cost over a series of estimates is double that of the standard computation. Also note that the accuracy of the estimate is identical to that of the non-distributed estimate.

The generalisation of the method to the case where the different sensors have different measurement accuracy can be performed as before.

The method is not strictly privacy preserving, as some intermediate information has leaked, *i.e.*, (15-20). This information would appear to be of little interest given a sufficient number (at least three) of non-colluding participants, as it does not reveal information about any individual sensor.

A. Tracking

Once again, we may consider the problem of tracking a moving target. Post-fusion tracking is in principle easy — we simply perform Kalman filtering on the position estimates. However, there is still the issue of estimating the measurement noise covariance. If each distance estimate D_i is an unbiased estimate of range d_i, with Gaussian errors of variance σ_i^2, then we can compute the covariance of the β given in (26) by

$$\text{Cov}(\beta) = A^{+}\text{Cov}(\mathbf{v})A^{+T}. \quad (30)$$

Assuming that the errors from each sensor are independent then $\text{Cov}(\mathbf{v})$ will be approximately diagonal with the terms along the diagonal ν_i^2 given by

$$\nu_i^2 = \text{Var}(v_i) \simeq \text{var}(D_i^2) = 2\sigma_i^2(\sigma_i^2 + 2d_i^2), \quad (31)$$

for large N and negligible errors in sensor positions. Note that we don't know d_i, but rather than using D_i, we may obtain a more accurate estimate from β. Given that $\text{Cov}(\mathbf{v})$ is diagonal

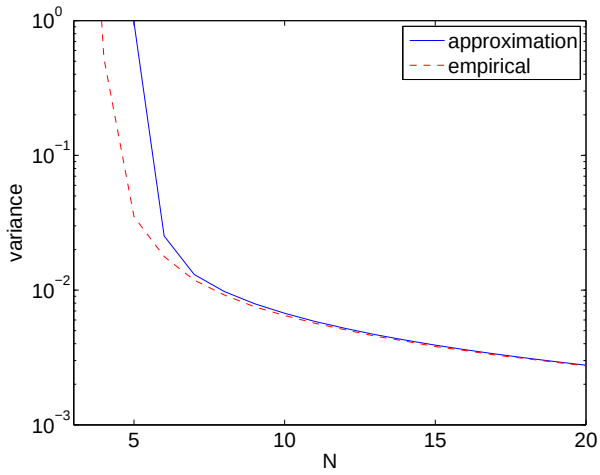
$$A^T\text{Cov}(\mathbf{v})A = \sum_{i=1}^N \begin{pmatrix} (X_i - \bar{X})^2\nu_i^2 & (X_i - \bar{X})(Y_i - \bar{Y})\nu_i^2 \\ (X_i - \bar{X})(Y_i - \bar{Y})\nu_i^2 & (Y_i - \bar{X})^2\nu_i^2 \end{pmatrix}. \quad (32)$$

We have already obtained $(A^T A)^{-1}$ from the process above, and so we can now derive an estimate of the covariance of the position estimate, again using a secure distributed summation for each component of $A^T\text{Cov}(\mathbf{v})A$, and we can use this covariance estimate in tracking without needing to share the variance of the distance estimates D_i.

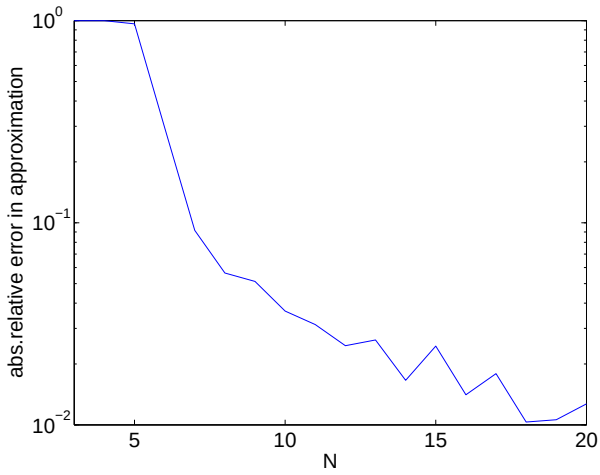
In the above approach there are three approximations: (i) that $\text{Cov}(\mathbf{v})$ will be approximately diagonal, (ii) that $\text{Var}(v_i) \simeq \text{var}(D_i^2)$ and (iii) that we can use $\hat{d}_i$ a distance estimate (based on the position estimate $(\hat{x}, \hat{y})$) in place of d_i in the calculation of covariance. We perform a simple simulation to test how well the approximation works. In the simulations we generate the target location and N sensor locations uniformly in the unit square, and simulate distance measures D_i with mean d_i and variance $\sigma^2 = 0.1$. We then perform the algorithm, and estimate the covariance as well as measuring the squared errors. We average both over 100,000 simulations to obtain estimates of the average covariance of the estimates. The cross-diagonal terms are close to zero, and so we focus on the diagonal elements. Due to symmetry, we only need examine the variance of the x co-ordinate estimate. Figure 2 shows the variance given by the approximation above in comparison to the empirical variance measured on the data. We can see that the approximation converges to the measured value as N increases, and that the approximation is reasonable for N larger than around 7. Given the input scenario, the measurement variance $\sigma = 0.1$ is rather large (notice that you also need $N \geq 7$ before the variance of the estimates becomes reasonably small), and so we also test a case with much smaller input variance $\sigma = 0.001$, and we find that the approximation converges more quickly, as shown in Figure 3.

As noted above the quantities (15-19), and hence (28) need only be computed once at the start of the co-operative effort (for stationary sensors), and the only parts that need be computed at each subsequent time interval are (29), and (32). Hence at each time step we can use these covariances as the measurement noise term R_t (which we now make dependent on time).

Pre-fusion tracking is much harder. We cannot track the position of the target from range information alone, and so each individual sensor cannot create a pre-fusion position estimate. However, each operator could use successive range estimates to form an estimate of radial velocities (using an extended Kalman filter), and combine these (similarly to the above) to form a Cartesian velocity estimates. We leave the exact form of the tracking algorithm for future work.

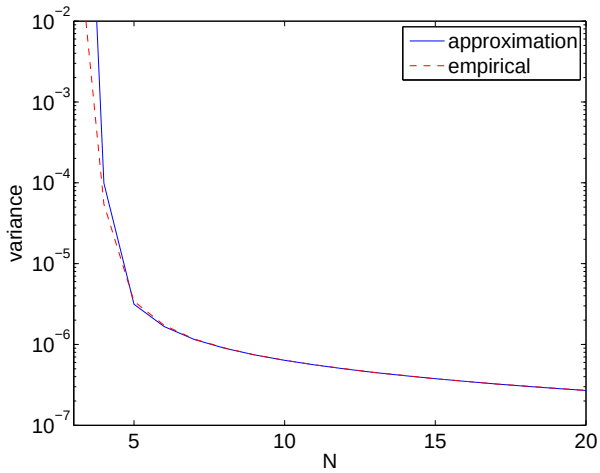


(a) Variance of the x co-ordinate measurement given variance of distance measurements of 0.1.

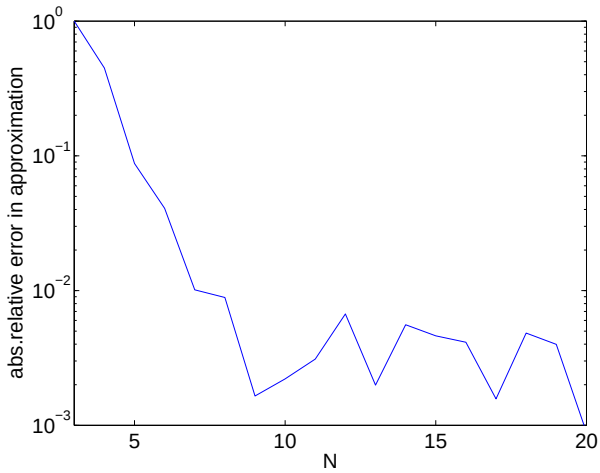


(b) The absolute value of the relative difference between the approximation and empirical measurements.

Fig. 2: A comparison of the approximate, and empirical variance estimates for localisation using N sensors for $\sigma_i = 0.1$.



(a) Variance of the x co-ordinate measurement given variance of distance measurements of 0.001.



(b) The absolute value of the relative difference between the approximation and empirical measurements.

Fig. 3: A comparison of the approximate, and empirical variance estimates for localisation using N sensors for $\sigma_i = 0.001$.

5. Conclusion

In this paper we make use of a number of techniques from the privacy preserving data mining literature to solve a novel problem in sensor fusion, namely that of localisation of a target using multiple sensors without revealing any particular information about the sensors' in question. In essence, we perform data fusion without data fusion, *i.e.*, we never bring the data together in one place, or reveal private information to other parties.

References

[1] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *Proc. of the ACM Workshop on Privacy in the Electronic Society (WPES'04)*, Washington, DC, USA, October 2004.

[2] J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Proc. Advances in Cryptology*, pp. 251–260, 1987.

[3] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT, LNCS*, pp. 236–252, 2005.

[4] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2), December 2002.

[5] N. Patwari, A. O. Hero, and J. A. Costa. *to appear*, chapter Learning Sensor Location from Signal Strength and Connectivity. Springer, 2006.

[6] M. Roughan and Y. Zhang. Secure distributed data-mining and its application to large-scale network measurements. *SIGCOMM Comput. Commun. Rev.*, 36(1):7–14, 2006.

[7] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *MobiHoc'03*, Annapolis, Maryland, 2003.

[8] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[9] A. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symposium on Foundations of Computer Science*, pp. 160–164, 1982.

[10] A. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science*, pp. 162–167, 1986.