Association Matrix Generation for Sensor Data Fusion

Michele Bierbaum
Johns Hopkins University/APL
11100 Johns Hopkins Road
Laurel, MD 20723-6099
USA
michele.bierbaum@jhuapl.edu

Abstract – *Accurate multi-sensor/multi-object data fusion is dependent on the accurate association of objects seen by one sensor system with those seen by another sensor system. Fusing data from N sensors on multiple objects can require the delineation of all possible association configurations among the observed N object groups. These configurations can be generated by first delineating all possible pair-wise associations between two sensors at a time, then combining these two-sensor associations into all possible N-sensor associations. This paper provides a combinatoric approach for enumerating and generating the two-sensor associations, with or without a priori association information, as well as a MatLab implementation of the algorithm. Additionally, an algorithm for combining the two-sensor associations into all possible N-sensor associations is described.*

Keywords: Object association, combinatorial algorithms, information fusion.

1 Introduction

In many fields – from medical imaging [1] to defense [2] to atmospheric and geo-sciences [3] – accurate fusion of data collected on multiple objects or source signals by multiple sensors is a common problem. Successful data fusion of this type generally involves the accurate association of objects among *N* object groups observed by *N* sensors, respectively. These *N*-sensor associations are a requirement in some fusion algorithms [4] as an initial step in the object association and data fusion process.

Using a combinatoric approach, an algorithm was developed and implemented to enumerate and generate all possible association configurations for two sensors, each sensor observing an arbitrary number of objects. These configurations are in the form of binary matrices with element values of 1's and 0's, assigning each object seen by one sensor to one or no object seen by a second sensor. These two-sensor associations are then combined into all possible *N*-sensor associations.

The basic enumeration algorithm for two-sensor association was briefly described in previous work [4]. The two-sensor association matrix generation algorithm, with or without *a priori* association information, its implementation in MatLab [5], and an algorithmic extension to *N*-sensor associations are additionally presented here.

2 Two-sensor association algorithm

Given a sensor $\mathcal{A}$ observing $a = 2$ objects and a sensor $\mathcal{B}$ observing $b = 3$ objects, a binary matrix of the form

$$C_3(a,b) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

represents one possible association scenario between the two object groups observed. The two-dimensional matrix notation of $C_i(a,b) \equiv C_i(\text{rows,columns})$ assigns each matrix row to an object seen by sensor $\mathcal{A}$ and each matrix column to an object seen by sensor $\mathcal{B}$. A pair-wise association of an $\mathcal{A}$ object with a $\mathcal{B}$ object is denoted by a "1" in the appropriate row and column. For example, in Eq. (1) sensor $\mathcal{A}$'s 1st object associates with sensor $\mathcal{B}$'s 1st object, sensor $\mathcal{A}$'s 2nd object associates with sensor $\mathcal{B}$'s 3rd object, and sensor $\mathcal{B}$'s 2nd object associates with no object observed by sensor $\mathcal{A}$. A complete association matrix set $\mathcal{C}(a,b)=\{C_1(a,b),C_2(a,b),\ldots,C_I(a,b)\}$ delineates all possible association hypotheses for the two object groups. Matrix C_3 in Eq. (1) represents the third matrix of the $\mathcal{C}(2,3)$ set. The generation of all possible two-sensor matrices $\{C_i(a,b)\}$ is a necessary first step in generating all possible associations among an arbitrary number of sensors.

The two-sensor association algorithm begins with the two sensors $\mathcal{A}$ and $\mathcal{B}$. Sensor $\mathcal{A}$ observes a objects and is represented by a set of a elements, numbered 1 through a: $A=\{1,2,\ldots,a\}$. Sensor $\mathcal{B}$ observes b objects and is represented by a set of b elements, numbered 1 through b: $B=\{1,2,\ldots,b\}$. Without loss of generality, assume $a \le b$. Thus, the following assumptions are made:

- $a \le b$
- an object seen by one sensor physically and uniquely corresponds to one or no object seen by the other sensor.

The combinatorial problem to solve is the unique pairing of elements of all subsets of A with elements of all subsets of B. The paired subsets must have the same number of elements. First, elements of the entire set A are uniquely paired with elements of a-element subsets of the set B. Next, elements of (a-1)-element subsets of A are uniquely paired with elements of (a-1)-element subsets of B. This

pairing of elements of subsets of A with elements of equally-sized subsets of B continues until the final pairing of a null or empty subset of A with a null subset of B. Enumerating these unique pairings of elements of all subsets of A with elements of all subsets of B yields a formula for the total number of possible association configurations or matrices, *I*. The derivation of this formula as a by-product of the algorithm follows.

The total number of *k*-element subsets of a *b*-element set is [6]

$$\frac{b!}{(b-k)!\,k!}. \tag{3a}$$

The total number of *k*-element subsets of an *a*-element set is

$$\frac{a!}{(a-k)!k!}. \tag{3b}$$

Pairing up each *k*-element subset of B with each *k*-element subset of A yields

$$\frac{a!}{(a-k)!k!}\cdot\frac{b!}{(b-k)!k!}. \tag{3c}$$

Next, permute one of these *k*-element subsets of B or A (but not both) to account for all possible unique pairings of elements between the k-element subsets. This permutation can be accounted for by multiplying either Eq. (3a) or Eq. (3b) by *k*!, which is the number of permutations for a *k*-element set [6]. Eq. (3c) then becomes

$$\frac{a!}{(a-k)!}\cdot\frac{b!}{(b-k)!}\frac{1}{k!}. \tag{4}$$

Eq. (4) is next summed over k=0,1,…,(a-1),a to account for all possible subset sizes. This final summation yields a formula that exhaustively enumerates all association configurations, i.e. all unique "pairings" of subsets of {1,2,…,b} with subsets of {1,2,…,a}:

$$I=\sum_{k=0}^{a}\frac{a!}{(a-k)!}\frac{b!}{(b-k)!}\cdot\frac{1}{k!}. \tag{5}$$

Applying this formula to the earlier example of sensor $\mathcal{A}$ observing a = 2 objects and sensor $\mathcal{B}$ observing b = 3 objects, the total number of association matrices is computed as

$$I=\sum_{k=0}^{2}\frac{2!}{(2-k)!}\frac{3!}{(3-k)!}\cdot\frac{1}{k!}=13. \tag{6}$$

An association matrix is created by a unique pairing of A and B subsets of the same length. The locations of the 1's in these matrices are found by pairing up, in order, elements of the subsets of interest. For example, the matrix C_3 in Eq. (1) – with rows indexed by sensor $\mathcal{A}$ objects and columns indexed by sensor $\mathcal{B}$ objects – was created by pairing elements of the two subsets, A subset {1,2} and B subset {1,3}. The first element of the A or "row" subset is 1 (i.e. 1st row), and the first element of the B or "column" subset is 1 (i.e. 1st column). Thus, the matrix element indexed by the 1st row and the 1st column is a 1. The second element of the A subset is 2 (i.e. 2nd row), and the second element of the B subset is 3 (i.e. 3rd column). The matrix element of the 2nd row and 3rd column is also a 1. Because there are no more elements in these subsets, the rest of the matrix elements are 0's, and C_3 of the matrix set $\mathcal{C}(2,3)$ has been created.

In the following section, a MatLab implementation of the two-sensor algorithm enumerates and generates all possible association matrices for arbitrary numbers of observed objects, a and b. Algorithm output is discussed in Sec. 4, followed by incorporation of *a priori* association knowledge in Sec. 5. Finally, an algorithm for expanding to N-sensor associations is discussed in Sec. 6.

3 MatLab implementation of algorithm

Given a objects observed by sensor $\mathcal{A}$ and b objects observed by sensor $\mathcal{B}$, the main function config found in the following subsection generates all possible $\{C_i(a,b)\}$ association configuration matrices. Additional functions called by config are found in subsequent subsections.

3.1 Main function (config)

```
function [Assoc]=config(A,B)

% calculates all association matrices, Assoc,
% for the event of one sensor observing A objects
% and a second sensor observing B objects

X = min(A,B);
Y = max(A,B);

mask = ones(A,B);

% to incorporate knowledge of prohibited pair-
% wise associations, enter new mask matrix
% here with 0's and 1's specifying forbidden and
% allowable associations, respectively.  For
% example, a 2x3 mask (i.e. sensor A sees 2
% objects and sensor B sees 3) imposing restric-
% tions of 1st A object cannot associate with the
% 1st or 2nd B object and 2nd A object cannot
% associate with 3rd B object has following form:
% mask = [0 0 1;1 1 0];

no_null = 0;

% if no_null above is set to 1, null associations
% are prohibited.

Total = total(A,B,no_null);

Assoc1=zeros(A,B,Total);
Assoc2=zeros(A,B,Total);
```

```
Total_ind = 0;

% first, generate all possible k subsets
% (for k=X,X-1,...1) of smaller set of objects

ind_obj = X;
while ind_obj ~= 0
    denom1 = factorial(ind_obj);
    denom2 = factorial(X-ind_obj);
    ind_final = factorial(X)/(denom1*denom2);
    min_ind_final = ind_final;
    M=zeros(ind_final,ind_obj);
    T=1:ind_obj;
    M(1,:)=T;
    ind = 1;
    while ind ~= ind_final
        ind=ind+1;
        M(ind,:)=S(T,ind_obj,X);
        T=M(ind,:);
      end

    % generate all possible subsets of larger set

    ind = 1;
    denom1 = factorial(ind_obj);
    denom2 = factorial(Y-ind_obj);
    ind_final = factorial(Y)/(denom1*denom2);
    U=zeros(ind_final,ind_obj);
    T=1:ind_obj;
    U(1,:)=T;
    while ind ~= ind_final
        ind=ind+1;
        U(ind,:)=S(T,ind_obj,Y);
        T=U(ind,:);
    end

    % permute these subsets

    ind=0;
    perm_final = ind_final*factorial(ind_obj);
    max_ind_final = perm_final;
    V=zeros(perm_final,ind_obj);
    while ind ~= ind_final
        ind=ind+1;
        index = ((ind-1)*(factorial(ind_obj)))+1;
        V(index,:)=U(ind,:);
        ind2 = 1;
        while ind2 ~= factorial(ind_obj)
            ind2 = ind2+1;
            T=V(index,:);
            index = (ind-1)*factorial(ind_obj);
            index = index + ind2;
            V(index,:)=P(ind_obj,T);
            T=V(index,:);
        end
    end
    N=V;

    % combine subsets into association matrices

    ind = 1;
    while ind ~= min_ind_final+1
      index = 1;
      while index ~= max_ind_final+1
          obj = 1;
          Total_ind = Total_ind + 1;
          while obj ~= ind_obj+1
              if A <= B
                  M1 = M(ind,obj);
                  N1 = N(index,obj);
                  T1 = Total_ind;
                  Assoc2(M1,N1,T1)=1;
              else
                  M1 = M(ind,obj);
                  N1 = N(index,obj);
                  T1 = Total_ind;
                  Assoc2(N1,M1,T1)=1;
              end
              obj=obj+1;
          end
          index=index+1;
      end
      ind=ind+1;
    end
    ind_obj=ind_obj-1;
end
```

```
% implement mask

mask_ind = 0;
for i=1:Total
    mask_flag=0;
    for a=1:A
        for b=1:B
            if Assoc2(a,b,i)==1 && ...
                    mask(a,b)==0
               mask_flag=mask_flag+1;
            end
        end
    end
    if mask_flag==0
       mask_ind=mask_ind+1;
       Assoc1(:,:,mask_ind)=Assoc2(:,:,i);
    end
end

Total = mask_ind
Assoc=zeros(A,B,Total);
for i=1:Total
    Assoc(:,:,i)=Assoc1(:,:,i);
end
```

3.2 Total number function (total)

```
function Number=total(A,B,no_null)

% given A objects and B objects, calculates
% total number of association matrices

X = min(A,B);
Y = max(A,B);

ind = 0;
Number = 0;

while ind ~= X+1
    frac1 = factorial(Y)/factorial(Y-ind);
    frac2 = factorial(X)/factorial(X-ind);
    Number = Number+frac1*frac2/factorial(ind);
    ind = ind + 1;
end

if no_null == 1
    Number = (factorial(Y)/(factorial(Y-X)));
end
```

3.3 Subset function (S)

```
function [F]=S(T,k,n)

% generate all k-subsets of list n

F=T;
i=k;
while (i >= 1) & (T(i) == n-k+i)
    i = i-1;
end

if i ~= 0
    j = i;
    while (j ~= k+1)
        F(j) = T(i) + 1 + j - i;
        j = j+1;
    end
end
```

3.4 Permutation function (P)

```
function [x] = P(n,x)

% permute n elements

i=n-1;
while x(i+1) < x(i)
   i = i-1;
end

if i ~= 0
    j = n;
    while x(j) < x(i)
```

```
        j = j - 1;
    end
    t = x(j);
    x(j) = x(i);
    x(i) = t;
    for h=(i+1):n,
        q(h) = x(h);
    end
    for h=(i+1):n,
        x(h)=q(n+i+1-h);
    end
end
```

4 Two-sensor algorithm output

Within the same MatLab directory, the above four functions are placed in their corresponding m-files. The main function config can be called from the MatLab prompt or from another m-file created by the user.

Continuing the example of sensors $\mathcal{A}$ and $\mathcal{B}$ observing 2 and 3 objects, respectively, the following lines can be placed in an m-file or entered line-by-line at the prompt.

```
A=2;
B=3;
Assoc=config(A,B)
```

The MatLab response is the total number of possible association matrices, followed by a listing of all the matrices, i.e. matrix set $\mathcal{C}(2,3)$:

```
Total =

    13


Assoc(:,:,1) =

     1     0     0
     0     1     0


Assoc(:,:,2) =

     0     1     0
     1     0     0


Assoc(:,:,3) =

     1     0     0
     0     0     1


Assoc(:,:,4) =

     0     0     1
     1     0     0


Assoc(:,:,5) =

     0     1     0
     0     0     1


Assoc(:,:,6) =

     0     0     1
     0     1     0


Assoc(:,:,7) =

     1     0     0
     0     0     0


Assoc(:,:,8) =

     0     1     0
     0     0     0


Assoc(:,:,9) =

     0     0     1
     0     0     0


Assoc(:,:,10) =

     0     0     0
     1     0     0


Assoc(:,:,11) =

     0     0     0
     0     1     0


Assoc(:,:,12) =

     0     0     0
     0     0     1


Assoc(:,:,13) =

     0     0     0
     0     0     0
```

Some object association algorithms require analyzing each matrix element in order to isolate object pairs of a particular association configuration, e.g. to compute distances (metric or statistical) between an object pair's positions in space (physical or feature) [4]. Matrix elements can be isolated by entering at the Matlab prompt or placing in an m-file created by the user:

```
Assoc(<row #>,<column #>,<matrix #>)
```

For example, the following command generates the value of the matrix element found in the 2nd row and 3rd column of the 5th matrix:

```
Assoc(2,3,5)
```

yielding the expected output for this matrix element:

```
ans =

     1
```

5 *A priori* association information

The algorithm and implementation just described is exhaustive in its enumeration and generation of all possible two-sensor association scenarios. As can be seen in Fig. 1, the computational complexity of this algorithm can be prohibitive without prior association knowledge to reduce the number of possible association scenarios. Such knowledge can be incorporated in the algorithm in one of two ways. The first is by the mask matrix variable `mask` in config. This matrix has *a* rows and *b* columns and can impose known restrictions on possible pair-wise associations between the two object sets. The element values of 0's and 1's in the matrix represent forbidden and allowable associations, respectively. Continuing the

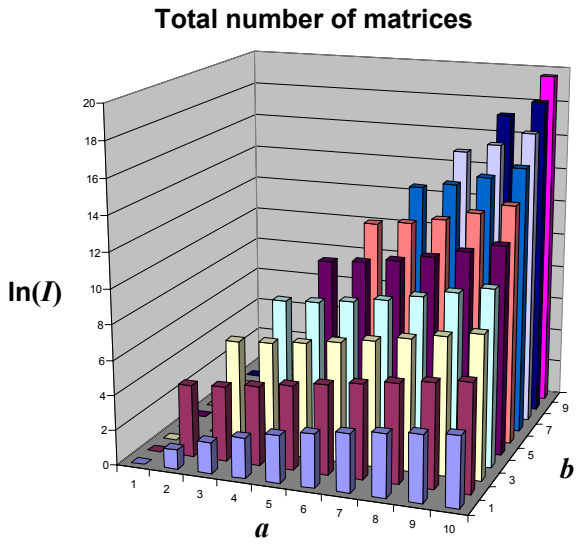

Fig. 1. Natural log dependence of total number of association matrices on increasing numbers of a and b objects [4].

example of sensor $\mathcal{A}$ seeing $a = 2$ objects and sensor $\mathcal{B}$ seeing $b = 3$ objects, the mask matrix M(a,b) prohibiting the 1st $\mathcal{A}$ object from associating with the 1st or 2nd $\mathcal{B}$ objects and the 2nd $\mathcal{A}$ object from associating with the 3rd $\mathcal{B}$ object is

$$\mathrm{M}(a,b) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}. \tag{7}$$

Entering this `mask` matrix in config in the form

```
mask = [0 0 1;1 1 0];
```

and running the algorithm again for $a = 2$ and $b = 3$ yields:

```
Total =

     6


Assoc(:,:,1) =

     0     0     1
     1     0     0


Assoc(:,:,2) =

     0     0     1
     0     1     0


Assoc(:,:,3) =

     0     0     1
     0     0     0


Assoc(:,:,4) =

     0     0     0
     1     0     0


Assoc(:,:,5) =

     0     0     0
     0     1     0


Assoc(:,:,6) =

     0     0     0
     0     0     0
```

A second way to incorporate prior association knowledge is by the `no_null` variable in config. This variable can impose known restrictions on null associations in general. Prohibiting null associations is equivalent to requiring that the minimum number of pair-wise object associations in each association configuration is equal to a, where $a \leq b$ from the algorithm development in Sec. 2. When prior knowledge suggests null associations are prohibited, the value of the `no_null` variable in config can be changed from its default setting of 0 to 1. Setting `no_null`=1 and running the algorithm using the values $a = 2$ and $b = 3$ yields the following matrices associating at least $a = 2$ objects from each sensor object group:

```
Total =

     6


Assoc(:,:,1) =

     1     0     0
     0     1     0


Assoc(:,:,2) =

     0     1     0
     1     0     0


Assoc(:,:,3) =

     1     0     0
     0     0     1


Assoc(:,:,4) =

     0     0     1
     1     0     0


Assoc(:,:,5) =

     0     1     0
     0     0     1


Assoc(:,:,6) =

     0     0     1
     0     1     0
```

6 Algorithmic extension to N sensors

The two-sensor algorithm can easily be extended to N-sensors, with each sensor observing an arbitrary number of objects. First, delineate all possible two-sensor pairs out of the N sensors, the total number of which is

$$K = \frac{N!}{(N-2)!2!}. \tag{8}$$

For each of these K sensor pairs generate all possible association matrices with the two-sensor algorithm in Secs. 2 and 3, taking into account any prior association information applicable to the sensor pair under consideration. For example, given four sensors $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$, observing a, b, c, and d objects, respectively, $N = 4$ and $K = 6$. Thus, there are six matrix sets of two-sensor associations to be created by the two-sensor algorithm, using any prior information available: $\mathcal{C}(a,b)$, $\mathcal{C}(b,c)$, $\mathcal{C}(c,d)$, $\mathcal{C}(d,a)$, $\mathcal{C}(a,c)$, and $\mathcal{C}(b,d)$.

Next, select one matrix from each two-sensor association matrix set $\mathcal{C}$, for a total of K matrices. These K matrices together define one possible N-sensor association scenario. Continuing with the example of the four sensors $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$, choose one matrix from each of the six $\mathcal{C}$ matrix sets to define one possible four-sensor association scenario:

$$\begin{Bmatrix} \mathrm{C}_2(a,b), & \mathrm{C}_4(b,c), & \mathrm{C}_2(c,d), \\ \mathrm{C}_8(d,a), & \mathrm{C}_1(a,c), & \mathrm{C}_5(b,d) \end{Bmatrix}. \tag{9}$$

Each N-sensor association scenario is then tested for validity by a series of matrix multiplications and comparisons among the K matrices defining the scenario.

First, group these K matrices into sets of three, in which each matrix in each set has one sensor in common with another matrix in the same set. The total number L of these sets is

$$L = \sum_{j=1}^{N-2} j(N-1-j). \tag{10}$$

For the four-sensor example, $N = 4$ resulting in $L = 4$ matrix sets of three derived from the matrices in Eq. (9):

$$\begin{gathered} \{\mathrm{C}_2(a,b), \mathrm{C}_4(b,c), \mathrm{C}_1(a,c)\} \\ \{\mathrm{C}_2(a,b), \mathrm{C}_5(b,d), \mathrm{C}_8(d,a)\} \\ \{\mathrm{C}_1(a,c), \mathrm{C}_2(c,d), \mathrm{C}_8(d,a)\} \\ \{\mathrm{C}_4(b,c), \mathrm{C}_2(c,d), \mathrm{C}_5(b,d)\} \end{gathered} \tag{11}$$

Next, select one of these L matrix sets and multiply two of the three matrices in this set – transposing one of these two matrices if necessary so that the objects of the sensor in common are assigned to the rows of the first matrix and to the columns of the second. Compare all elements in the product matrix P with those in the third or "unused" matrix C – also transposed if necessary so that the sensors "assigned" to its rows and columns are the same as those for the P matrix. If the P matrix has a "1" in any matrix element location where the C matrix has a "0", this is an association violation as illustrated in Fig. 2. Only one violation is necessary to invalidate the K matrices of interest as a possible N-sensor association. If there is no violation in this product, a multiplication of a second

$$\mathrm{P} = \begin{bmatrix} 0 & 0 & \textcircled{1} \\ 0 & 1 & 0 \end{bmatrix} \quad \Updownarrow \quad \mathrm{C} = \begin{bmatrix} 1 & 0 & \textcircled{0} \\ 0 & 1 & 0 \end{bmatrix}$$

Fig. 2. Example of a one-to-one comparison of matrix elements in a product matrix P and a third matrix C. The association violation is circled.

combination of matrices in the L set under consideration is carried out and compared with the remaining third matrix. If there is again no violation, this process is carried out a third and final time for the set by multiplying the last possible two-matrix combination and comparing with the third matrix. If no association violations have been found in the entire set, the next L set is then tested, and so on and so forth. The N-sensor association configuration represented by the K matrices of interest is validated after all L sets have been tested with no violations found.

For example, in Eq. (11), suppose $a = 2$, $b = 3$, $c = 3$, and the matrices in the first L set are

$$\begin{aligned} \mathrm{C}_2(a,b) &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ \mathrm{C}_4(b,c) &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \\ \mathrm{C}_1(a,c) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{12}$$

Compare elements of $\mathrm{C}_1(a,c)$ with elements of the product matrix $\mathrm{P}(a,c)=[\mathrm{C}_2(a,b)\mathrm{C}_4(b,c)]$:

$$\mathrm{P}(a,c) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \Updownarrow \quad \mathrm{C}_1(a,c) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{13}$$

The P and C matrices in Eq. (13) are the same as those in Fig. (2). Thus, there is an association violation, and the four-sensor association scenario defined by the matrix set in Eq. (9) has been nullified, regardless of the matrices representing $\mathrm{C}_2(c,d)$ $\mathrm{C}_8(d,a)$, and $\mathrm{C}_5(b,d)$. However, if there had been no association violation in Eq. (13), the

matrix elements of $C_2(a,b)$ would next be compared with the elements of the product $P(a,b)=[C_1(a,c)C_4(b,c)^T]$. If this comparison also yielded no violations, the elements of $C_4(b,c)$ would next be compared with the elements of $P(b,c)=[C_2(a,b)^T C_1(a,c)]$. If this last comparison also resulted in no association violations, the first L set of Eq. (11) would have been successfully tested, and the second L set would next be tested in the same manner, and so on and so forth. The four-sensor association configuration defined by Eq. (9) would be validated only after all four L sets in Eq. (11) have been tested with no violations found.

Finally, select a different combination of K matrices – one from each two-sensor association matrix set $\mathcal{C}$ – defining another N-sensor association scenario to test for validity in the same manner. In terms of the four sensor example, choose a different set of K matrices from that in Eq. (9) to test. Repeat the entire process until all possible combinations of K matrices have been tested. The K-matrix combinations that are validated delineate all possible N-sensor association scenarios.

7 Conclusions

An algorithm for enumerating and generating association matrices for multiple objects observed by two sensors was described and implemented in MatLab. Incorporation of prior association knowledge into the algorithm as well as an extension of the algorithm to object associations among N sensors was also described. In some multi-sensor/multi-object association and data fusion processes, identifying the correct N-sensor association configuration among the N object sets observed is an important step. The algorithms described in this paper provide a crucial first step in identifying this correct association configuration through the generation of all possible association configurations for an N-sensor scenario, with and without prior association information.

Acknowledgements

I would like to thank Dr. Donald Maurer at JHU/APL for his helpful suggestions and proofreadings of this paper.

References

[1] Pedro Hojen-Sorensen, Lars Kai Hansen, and Ole Winther. Mean field implementation of bayesian ICA. *Neural Computation*, 14:889-918, 2002.

[2] Buddy H. Jeun and Allan Whittaker. Multi-sensor information fusion technology applied to the development of smart aircraft. *7th International Command and Control Research and Technology Symposium*, Quebec City, QC, Canada, 16-20 September 2003.

[3] S. Uselton, J. Ahrens, W. Bethel, L. Treinish, and A. State. Multi-source data analysis challenges. In *Proc. of the IEEE Computer Society Visualization '98*, pages 501-504, Raleigh, NC, October 1998.

[4] Michele M. Bierbaum and Robert L. Fry. Bayesian source separation and system data fusion methodology. In Robert L. Fry, editor, *Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 21st International Workshop (MaxEnt2001)*, pages 109–124, Baltimore, MD, 4-9 August 2001, AIP, Melville, NY, 2002.

[5] *MATLAB Reference Guide.* The Math Works, Inc., Natick, MA, 1996. http://www.mathworks.com/

[6] Donald Kreher and Douglas Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*. CRC Press, New York, NY, 1999.