A Comparison of Two Hypothesis Generation Algorithms in JPDAF Multiple Target Tracking

Pavлина Константинова, Кирп Алексиев

Abstract: Two algorithms for hypothesis generation in Multiple Target Tracking (MTT) with Joint Probabilistic Data Association Filter (JPDAF) have been developed and compared. The known Depth First Search (DFS) algorithm has been used as base approach. Some worst case examples have been used to compare effectiveness of the two program implementations in C++. The results proved that the recursive procedure has a little better performance than the iterative procedure with explicit stack.

Key words: Multiple Target Tracking (MTT), Joint Probabilistic Data Association (JPDA), Depth First Search (DFS) algorithm.

1. INTRODUCTION

There are many data association techniques used in MTT systems ranging from the simpler nearest-neighbour approaches to the very complex multiple hypothesis tracker (MHT). The simpler techniques are commonly used in MTT systems, but their performance degrades in clutter. The more complex MHT provides improved performance, but it is difficult to implement and in clutter environments a large number of hypotheses may have to be maintained, which requires extensive computational resources. Because of these difficulties, some other algorithms having smaller computational requirements were developed. These techniques are based on probabilistic data association (PDA) [1,2], that uses a weighted average of all the measurements falling inside a track's validation region. The Joint Probabilistic Data Association Filter (JPDAF) is the extension of the PDAF to the multitarget case. The JPDAF is the same as the PDAF except for the computation of the association probabilities. The measurement-to-track association probabilities are computed jointly across all targets and measurements. One essential part of this algorithm is hypothesis generation.

The goal of this paper is to study two implementations of the hypothesis generation procedures and to compare their effectiveness in order to choose the more appropriate for real time target tracking application.

2. PROBLEM FORMULATION

Let m and n are the number of measurements and targets, respectively, and k is the time index. The dynamic model [9] for target t is described by:

\[
x'(k+1) = F'(k)x'(k) + G'(k)w'
\]

\[
z'(k) = H'(k)x'(k) + v'(k), \quad t = 1,2,...,n
\]

where \(x'(k)\) is the \(N'\)-dimensional state vector, \(z'(k)\) is the \(M'\)-dimensional measurement vector with \(M'\) actually independent of \(t\), \(F'(k), G'(k)\) and \(H'(k)\) are known model matrices, \(w'(k)\) and \(v'(k)\) are respectively, the \(p'\) and \(M'\)-dimensional noise vectors, which are assumed to be zero-mean independent identically distributed Gaussian processes with known covariances.

Suppose that \(m\) measurements are received at time index \(k\). In a cluttered environment, \(m\) does not necessarily equal \(n\) and it may be difficult to distinguish whether a measurement originated from a target or from clutter. A validated measurement is one which is either inside or on the boundary of the validation gate of a target. Mathematically, a validation gate is defined by

\[
(z(k) - \hat{z}'(k))^T S'(k) \hat{z}'(k) \leq g^2,
\]

where \(\hat{z}'(k)\) is the predicted value of \(z'(k)\) for target \(t\). The error \((z(k) - \hat{z}'(k))\) is the innovation, generated from \(z(k)\) for target \(t\), \(S'(k)\) is its covariance matrix and \(g\) is a
selected threshold. The choice of \( g \) has to ensure that the correct measurements will lie within the gate with the specified probability. The inequality given in (3) is a validation test. The result of the test is kept in a validation matrix \( \Omega \). This validation matrix \( \Omega \) is a \( m \times (n + 1) \) rectangular matrix defined as [8]

\[
\Omega = \left[ \omega_{jt} \right] = \begin{bmatrix}
0 & 1 & 2 & \ldots & n \\
1 & \omega_{11} & \omega_{12} & \ldots & \omega_{1n} \\
1 & \omega_{21} & \omega_{22} & \ldots & \omega_{2n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega_{m1} & \omega_{m2} & \ldots & \omega_{mn} \\
\end{bmatrix}
\]

where \( \omega_{j0} = 1 \) means that measurement \( j \) could originate from clutter, \( \omega_{jt} = 1 \) if measurement \( j \) is inside the validation gate of target \( t \) and \( \omega_{jt} = 0 \) if measurement \( j \) is outside the validation gate of target \( t \) for \( j = 1, 2, \ldots, m \), and \( t = 1, 2, \ldots, n \). Based on the validation matrix, data association hypotheses (or feasible events [3,8]) are generated subject to the following two restrictions:

- each measurement can have only one origin (either a specific target or clutter)
- no more than one measurement originates from a target

To reduce the computational cost for data association, we will discuss one of the essential parts of the JPDAF algorithm, namely hypotheses generation.

### 2.1 Choosing hypothesis generation approach

In [8] it is pointed out that in the context of tracking multiple targets in clutter, data association can be modelled as an exhaustive search problem. Because the hypothesis generation process could be represented like a tree, we look for appropriate algorithm for search in the tree. According to whether there is enough information for ordering choices, search methods can be roughly classified into two kinds [6] as is shown in fig. 1. The most appropriate for solving hypotheses generation problem is Depth First Search algorithm (DFS). For the DFS algorithm implementation two procedures are proposed in [6]:

- Using recursion
- Using explicit stack

![Fig.1 Search Methods Classification](image)

In many cases it is better avoiding recursion if there is iterative solution [7].

### 2.2 Hypotheses tree

The set of all possible hypotheses for given scenario can be represented as a tree. For a little example of 2 tracks and 3 measurements, if in the first track's gate fall 1-st
and 2-nd measurements, and in the second track’s gate fall 2-nd and 3-rd measurements there are 8 hypotheses. They are illustrated in fig. 2.

\[
\varepsilon_0(0,0,0) \quad \text{Level 0}
\]

\[
\varepsilon_1(1,0,0) \quad \varepsilon_4(0,1,0) \quad \varepsilon_6(0,2,0) \quad \varepsilon_7(0,0,2) \quad \text{Level 1}
\]

\[
\varepsilon_2(1,2,0) \quad \varepsilon_3(1,0,2) \quad \varepsilon_5(0,1,2) \quad \text{Level 2}
\]

Fig.2 Hypotheses tree

2.3. Computational complexity of the algorithm in Multiple Target Tracking

Application

In the worst case when all \( m \) measurements fall in the intersection of the validation regions of all \( n \) tracks, the number of hypotheses can be obtained as:

\[
S(n,m) = \sum_{i=0}^{\min(n,m)} C_m^i A_n^i, \quad \text{where} \quad (5)
\]

\[
C_m^i = \frac{m!}{i!(m-i)!} \quad \text{for} \quad 0 \leq i \leq m \quad \text{and} \quad A_n^i = \frac{n!}{(n-i)!} \quad \text{for} \quad 0 \leq i \leq n \quad (6)
\]

With these formulae the number of hypotheses for various values of the \( m \) and \( n \) are computed and are shown later in the table 1 of results. The enormous increasing of the number of hypothesis can be seen.

3. MATHEMATICAL MODEL FOR DATA ASSOCIATION

The proposed in [8] mathematical model is based on the following typical problem description: There exist \( m \) variables \( X_j (j = 1,2,..,m) \). The value of each of them \( X_j \) belongs to a set \( Z_j \), where \( Z_j \) is a finite and linearly ordered. A candidate solution for this problem is a set of values \( (X_1, X_2,...,X_m) \).

The objective here is to find either one solution or all solutions under the imposed constraints. An efficient algorithm to solve the above problem is DFS procedure

In the context of tracking multiple target in clutter, the problem of data association can be modelled as an "exhaustive search problem" with a set of proper notations. Let \( X_j (j=1,2,...,m) \) denotes measurement \( j \). The value of \( X_j \) identifies the target or the clutter which is hypothesised to be associated with measurement \( j \). For example \( X_j = 3 \) implies that measurement \( j \) is hypothesised to be associated with track number 3, and \( X_j = 0 \) denotes that measurement \( j \) is hypothesised to be associated with clutter. Therefore the set \( Z_j \) may be defined by

\[
Z_j = \{ t | \omega_{jt} = 1 \}, \quad j = 1,2,...,m \quad \text{and} \quad t = 0,1,2,...,n
\]

Note that since \( \omega_{j0} \) is equal to 1, therefore, \( 0 \in Z_j \) for \( j = 1,2,...,m \). If measurement \( j \) falls inside the validation gate of target \( t \), then \( t \in Z_j \).

The two constraints, which have to be satisfied for a feasible event, can be easily translated into the language of the exhaustive search problem for data association. Hence an \( m \)-tuple, \( (X_1, X_2,...,X_p,...,X_q,...,X_m) \) is a solution if the following two constraints are satisfied:
1) If \( p \neq q, X_p \neq 0 \) and \( X_q \neq 0 \), then \( X_p \neq X_q \).

2) If \( X_p = X_q \) and \( p \neq q \), then \( X_p = X_q = 0 \).

4. HYPOTHESES GENERATION ALGORITHM DESCRIPTION

There are some differences between the known DFS algorithm for exhaustive search in the tree and algorithm for the generation of data association hypotheses. In the first case usually there are no solutions that are known in advance, but in the problem of data association always known in advance is \((0,0,...,0)\). The remaining solutions can be generated systematically from various valid combinations of non-zeroes values of the elements. In order to illustrate this idea consider again example in the previous sections in fig.2. The root of the tree is the starting solution \((0,0,0)\) when all measurements are hypothesised to be false alarms. Each node at level \( i \) is associated with \( i \) nonzero variables, in corresponding set of values. In general the height of the tree is \( \min(m,n) \) since \( i \) must be less than or equal to both \( n \) and \( m \). The explicit stack can be implemented as an array [5]. The stack \( \hat{X} \) contains the set of defined values \( X_{j1}, X_{j2} \) and \( X_{jL} \) which represent the current hypothesis.

![Flowchart of DFS procedure for data association](image)

The function GetNext is logical and returns true if the imposed constraints are satisfied. When all positions at a given level are checked the level has to be reduced (so called "backtracking"), to get next element from the current level.

5. RESULTS

The results from the two implementations are given in table 1 and shows that the recursive procedure has a little better performance.

**Table 1.** Elapsed time in seconds for the implemented worst case examples using: 1) recursion and 2) explicit stack.

<table>
<thead>
<tr>
<th>Hypothesis number</th>
<th>n=2, m=2</th>
<th>n=3, m=5</th>
<th>n=4, m=10</th>
<th>n=4, m=12</th>
<th>n=5, m=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>t[sec] for recursion</td>
<td>0</td>
<td>0</td>
<td>0.017</td>
<td>0.036</td>
<td>0.166</td>
</tr>
<tr>
<td>t[sec] for explicit stack</td>
<td>0</td>
<td>0</td>
<td>0.021</td>
<td>0.043</td>
<td>0.196</td>
</tr>
</tbody>
</table>
In practical implementation [7] it is especially important that the recursive depth have to be both finite and sufficiently little number. The depth of the recursion will be no more than the height of the tree, which depends on the number of tracks and the number of measurements i.e. depends on scenario. In cluttered environment there is possibility for stack overflow. From these considerations the conclusion is that it is reasonable to use the iterative procedure as more reliable and sufficiently effective.

CONCLUSIONS

Many real world problems can usually be modelled and transformed to some searching problem. The hypothesis generation in MTT systems with JPDA Filter also can be considered as problem of this sort. Two program implementations for hypotheses generation are realised in C++ code. The first one uses recursion and the second uses explicit stack. The elapsed times of the both implementations are compared. From the results it is seen that the elapsed time for recursive procedure is a little better than the time for iterative procedure with explicit stack. For the target tracking problem the depth of the recursion will be no more than the height of the hypotheses tree, which depends on scenario. To assure more reliable program work in cluttered environment when the depth of the recursion may be not sufficiently little we conclude that using the iterative procedure with explicit stack is more suitable.

REFERENCES


ABOUT THE AUTORS

Dr. Pavlina Konstantinova, PhD, senior researcher at Central Laboratory for Parallel Processing, BAS, Sofia, Phone: +359 979-66-20, E-mail: pavlina@bas.bg.
Dr. Kiril Alexiev, PhD, senior researcher at Central Laboratory for Parallel Processing, BAS, Sofia, Phone: +359 979-66-20, E-mail: alexiev@bas.bg