MHT-X: offline multiple hypothesis tracking with algorithm X

Peteris Zvejnieks1 · Mihails Birjukovs1 · Martins Klevs1 · Megumi Akashi2 · Sven Eckert2 · Andris Jakovics1

Received: 15 September 2021 / Revised: 21 December 2021 / Accepted: 7 February 2022 / Published online: 8 March 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
An efficient and versatile implementation of offline multiple hypothesis tracking with Algorithm X for optimal association search was developed using Python. The code is intended for scientific applications that do not require online processing. Directed graph framework is used and multiple scans with progressively increasing time window width are used for edge construction for maximum likelihood trajectories. The current version of the code was developed for applications in multi-phase hydrodynamics, e.g., bubble and particle tracking, and is capable of resolving object motion, merges and splits. Feasible object associations and trajectory graph edge likelihoods are determined using weak mass and momentum conservation laws translated to statistical functions for object properties. The code is compatible with n-dimensional motion with arbitrarily many tracked object properties. This framework is easily extendable beyond the present application by replacing the currently used heuristics with ones more appropriate for the problem at hand. The code is open-source and will be continuously developed further.

Graphical abstract
1 Introduction

Experiments with multi-phase flow often involve rather complex flow patterns, especially the turbulent dispersion of particles, bubbles, droplets, etc. It is of particular interest to track these objects and their interactions to reconstruct their dynamics and interpret the underlying physics such as collisions, breakup, coalescence, agglomeration, coherence of motion and shape evolution. In cases with greater number densities of objects and/or when object positions and properties originate from noisy measurements, reconstructing object motion can be a significant challenge. This is because noise in experimental data often leads to false positives and missing and/or distorted data.

Multiple hypotheses tracking (MHT) is classically considered to be the most reliable method for finding the optimal solution for data association problems (Kim et al. 2015; Rubio et al. 2012; Blackman 2004; Cox and Hingorani 1996; Reid 1979). However, it is rarely used so far for real-time (online) applications due to its computational complexity. However, it is rarely used so far for real-time (online) applications due to its computational complexity (Knuth 2000). This way the computational complexity becomes consistently sub-exponential, allowing to cover the entire search-space of viable solutions, while entirely omitting contradictory solutions.

The tracking algorithm described herein was developed in Python for the analysis of multi-phase hydrodynamic systems, such as bubble and particle flow in liquids. In our field of research, there are several cases of interest where explicit, precise and robust object tracking is desired: dynamic optical, X-ray and neutron imaging of argon bubble flow in liquid gallium or GaInSn eutectic alloy (Birjukovs et al. 2021, 2020a, b; Akashi et al. 2019; Obiso et al. 2020; Keplinger et al. 2019, 2018); neutron imaging of gadolinium oxide particle flow in liquid gallium (Lappan et al. 2020; Sarma et al. 2015; Ščepanskis et al. 2017; Dzelme et al. 2018); optical imaging of salt crystals and liquid crystal tracers in water (Anders et al. 2019, 2020); neutron imaging of gadolinium particles in froth (Heitkam et al. 2019); bubble flow simulations (Birjukovs et al. 2020a, b; Obiso et al. 2020). In all of these cases, the objective is to trace objects (bubbles/particles segmented beforehand) in time, reconstructing their trajectories, resolving merging and splitting events (i.e., bubble coalescence and particle adhesion) where these occur and performing particle tracking velocimetry (PTV). The output can then be used for an in-depth analysis of bubble/particle collective dynamics, comparison of simulation and experimental data, etc. The code was also designed to be able to account for significant shape oscillations and parameter variations for objects that are typical in the above applications.

While this means that the utilized heuristics such as association probability functions and tracing examples showcased herein are mostly applicable to the above cases, the main framework described in this paper is universal and can be readily adapted to any type of offline tracking problem, abstract or physical. Most modern tracking algorithms are designed for use on live data (military, surveillance, transportation and other related applications) (Kim et al. 2015), whereas here the offline implementation is dictated by scientific use, where real-time data interpretation is not required, but rather reliable solutions are expected for potentially very large scale problems within acceptable time intervals (i.e., not necessarily requiring high performance computing).

Directed graph representation for input data and trajectories was chosen because of its intuitive interpretation, which makes the presented tracing framework conceptually easier to implement and customize for the application at hand (Rubio et al. 2012). This MHT implementation, referred to as MHT-X in this paper, consists of several core components, in this case Python objects, where each object represents a stage/method for finding the optimal solution for the tracking problem—the objects and methods are covered in the following section.

2 The algorithm

The algorithm starts off with an initialization step, where the data is used to construct an initial graph. The algorithm then proceeds to scan the graph over time (t) with a time window (i.e., a time interval over which it is attempted to resolve associations between objects) of an iteratively increasing width (∆t). This is done to progressively resolve object associations over different timescales. Solutions are
then stored on directed graph edges, where nodes represent object detections.

Each time a new $\Delta t$ value is set, unlikely edges are eliminated and a graph sweep over time is executed, where at every time step the following operations are performed:

1. Objects (trajectories) with unresolved endpoints are found
2. Associations are formed for these objects
3. Optimal hypotheses for associations are identified
4. The optimal solution is added to the graph

Initially $\Delta t = 1$ (two neighboring frames are considered) and it is incremented up to a value for which the utilized predictive models of object motion are no longer reliable, i.e., when tracking yields results with unphysically abrupt trajectory cuts, jumps and/or oscillations.

A schematic representation of the framework described above can be seen in Fig. 1.

2.1 Input and measurements

Since the tracking algorithm is offline, a complete set of measurements for object detection events is required as input. For each detection event an object $O_{t_i}$, where $t$ is the time index, indicating detection time (frame number) and an $i$ is an auxiliary trajectory index, indicating which trajectory $O_{t_i}$ belongs to. It is auxiliary in that it is used herein for mathematical formulation, but is not utilized in the implemented code. $t, i \in \mathbb{N}$ where $t \in [1, N]$ and $N$ is the number of consecutive frames in the input dataset. Each $O_{t_i}$ is also prescribed a set of properties, which in this case is $\{\vec{r}, S\}$ where $\vec{r}$ is the radius vector for the object centroid and $S$ is the projection area due to optical/x-ray/neutron transmission.

Generally, however, there are no restrictions on the type and number of properties, but rather this is dictated by the intended application of the present framework.

2.2 Directed graph architecture

The time-forward directed graph representation of data is the natural framework for MHT and specifically for the problem at hand, and makes the data association formation effortless, since instead of explicitly constructing trajectories, associations are defined between data points and from these associations one can construct trajectories as necessary.

A directed graph $G$ is defined by a set of nodes $V$, a set of edges $E$ and a function $P$. The node set consists of the detected objects and two special nodes ($V^*$). The special nodes are labeled $\text{Entry}$ and $\text{Exit}$ which are auxiliary (padding) nodes of the graph that represent start and end points of trajectories, respectively. Special nodes do not possess spatial or temporal locations. Directed edges of this graph are represented by an unordered set $E$ of ordered pairs of elements of $V$. The function $P$ maps every edge in the graph to its respective likelihood value $p$.

\[
\begin{align*}
V &= \{O_{t_i}\} \cup V^* \\
E &= \{(a, b)\} \quad a, b \in V, a \neq b, \{a, b\} \neq V^*, a \neq \text{Exit}, b \neq \text{Entry} \\
P : E \rightarrow p & \quad p \in (0, 1] \\
G &= (V, E, P)
\end{align*}
\]

(1)

Edges represent associations between the nodes and are assigned likelihood values to which $P$ maps every edge. Likelihood represents the probability that the association is true. Node associations represented by directed edges are aligned with the arrow of time. Edges connecting graph nodes to the special nodes are used to store the likelihoods.

![Flowchart of the developed tracing algorithm MHT-X](image-url)
that the connected trajectory nodes are trajectory endpoints. Multiple edges directed away from or towards a node that is not a special node represent split/merge events, respectively, and each of the edges from one such event has an equal likelihood value, which represents the likelihood of that event. The directed graph formed by this algorithm represents different trajectories encompassed by it where trajectories are potentially logically connected via split/merge events. A schematic representation of a graph formed after tracing is shown in Fig. 2 and its structure is described in more detail below.

2.3 Trajectories

A trajectory is a set of sequentially connected objects that were measured over different time frames, which in this framework translates to sets of nodes connected by directed edges, bounded by special nodes and/or split or merge events. In cases where one-to-many split or many-to-one merge events occur, the source trajectories are broken and new ones are formed, generating a set of logically connected trajectories – trajectory families. Families are still bounded by special nodes. Nodes which are only connected to the special nodes are considered false detections. The latter is not a design feature, but rather it is an emergent property of the framework.

The set of all trajectories is defined as follows:

\[
T = \left\{ T_k \mid T_k = \{O_{tk}\}, t \in [t_1:t_k], k \in \mathbb{N} \right\}
\]  

(2)

Given a trajectory, one may also define functions that map trajectories to other mathematical objects: this way, one can derive velocity and acceleration over trajectories, as well as use trajectory data to make predictions regarding object motion – all of this can be used to define heuristics for more efficient object tracking.

To extrapolate a trajectory reliably, a quantitative model is necessary. For MHD multi-phase flow, this version of the code utilizes a rather naive general approach for extrapolation: a spline operator \( \hat{S} \) is defined, which maps a trajectory to a piece-wise polynomial function \( g_k \) of time

\[
\hat{S} : T_k \rightarrow g_k(t)
\]  

(3)

This spline is then used for extrapolation as follows:

\[
T_k(t|t > t_k) = g_k(t_k) + \frac{d}{dt} g_k(t - a) \cdot (t - t_k)
\]

\[
T_k(t|t < t_1) = g_k(t_1) + \frac{d}{dt} g_k(t + a) \cdot (t - t_1)
\]  

(4)

Where \( a \) is an arbitrary constant used to compensate the fact, that splines go trough the last point at an arbitrary angle.

2.4 Initialization

During the initialization step the nodes of \( G \) are defined from the set of measurements. A critical step here is to generate boundary conditions (constraints). It is known \( a \) priori that all nodes with \( t = 1 \) or \( t = N \) must be the endpoints of trajectories. As an initial condition for the graph, all nodes in the graph and the special nodes are interconnected, and each of the edges is prescribed a likelihood value based on the following Boolean statements. The set of edges formed in this step formally is defined as follows:

\[
E_1 = \left\{ (\text{Entry}, O_{t_1}) \mid \forall O_{t_1}, P : (\text{Entry}, O_{t_1}) \rightarrow \left\{ \begin{array}{ll} 1, & t = 1 \\ 0, & t \neq 1 \end{array} \right\} \right\}
\]

\[
E_2 = \left\{ (O_{t_1}, \text{Exit}) \mid \forall O_{t_1}, P : (O_{t_1}, \text{Exit}) \rightarrow \left\{ \begin{array}{ll} 1, & t = N \\ 0, & t \neq N \end{array} \right\} \right\}
\]

\[
E = E_1 \cup E_2
\]  

(5)

This step is necessary so that nodes in either the first or the last frame have unambiguous origins/continuations and ensures that these connections are not be broken during the edge eradication step.

2.5 Graph and time window width sweeps

The graph sweep is the main part of the algorithm—it iteratively performs hypotheses definition and evaluation, optimal association search and graph edge insertion using subroutines described later in this section. For this, a time window of width \( \Delta t \) is defined, i.e., the number of frames covered by the window is \( \Delta t + 1 \). This window is then translated forward in time frame-by-frame through the graph inserting new edges within each time window where appropriate. Before this graph sweep is executed, a fully connected graph is expected, i.e., every node (except the special nodes) must have incoming and outgoing edges.
After this initial graph is formed, a fraction of its edges are removed using the edge eradication method. This yields a sparsely connected graph where some nodes have no incoming and/or outgoing edges. These ambiguities are resolved during the graph sweep yielding a fully connected graph that can be taken as the solution to the tracking problem or can be used as an initial condition for a new graph sweep with a different time window width.

In the current implementation this is done repeatedly, constituting a time window width sweep. After an initial fully connected graph is formed and edge eradication is invoked, Δt = 1 is set defining a 2-frame window with \( t \in [t + \Delta t] \). A graph sweep with this window resolves the most obvious associations formed between sequential frames. The resulting graph, after unlikely edge eradication, is used as an initial condition for the next graph sweep with a 3-frame interrogation window (Δt = 2). This process is repeated with increasing Δt until it exceeds a user defined threshold. This is done so that consecutive graph sweeps resolve associations over greater distance in time, allowing to overcome detection failures. The following subsections describe the underlying subroutines in detail.

2.5.1 Unlikely edge eradication

After every graph sweep and initialization, all measurement nodes will have incoming and outgoing edges. New initial conditions for the next time sweep are generated by removing unlikely edges, which are identified as follows. Using the \( p \) distribution for all of the graph edges and a user defined quantile parameter \( q \in [0, 1] \), a likelihood threshold \( p_c \) is computed. Edges with \( p \leq p_c \) are eradicated.

2.5.2 Sets of associable trajectories

At every time step in the graph sweep for a given \( t \) and \( \Delta t \) two unordered sets of trajectories are formed: \( F_t \) — a set of trajectories which do not have an endpoint within \( [t:t + \Delta t] \); \( B_t \) — a set of trajectories which do not have an endpoint within \( (t:t + \Delta t] \). Formally \( F_t \) and \( B_t \) are defined as

\[
F_t = \bigg\{ (T_k) \bigg| t_k \in [t:t + \Delta t] \land \nexists (O_{i,k}, \forall O_{h_i}) \bigg\} \quad (6)
\]

\[
B_t = \bigg\{ (T_k) \bigg| t_k \in (t:t + \Delta t] \land \nexists (\forall O_{h_i}, O_{i,k}) \bigg\} \quad (7)
\]

where \( i \in [1:N] \).

2.5.3 Associations

The types of associations \( A \) supported by the current version of the tracer are formally defined as follows:

\[
\begin{align*}
\text{Entry} & \rightarrow (\emptyset, T_k), T_k \in B_t \\
\text{Exit} & \rightarrow (T_k, \emptyset), T_k \in F_t \\
\text{Translation} & \rightarrow (T_k, T_m), T_k \in F_t \land T_m \in B_t \\
\text{Split} & \rightarrow (T_k, \{T_m\}), T_k \in F_t \land \forall T_m \in B_t \\
\text{Merge} & \rightarrow ([T_k], T_m), \forall T_k \in F_t \land T_m \in B_t \\
\end{align*}
\]

The associator routine in the code takes an element of \( F_t \) and using the pairwise association condition determines which of the elements of \( B_t \) can be associated with it. This results in a set of elements associable to \( F_t \). Because each \( F_t \) element can be associated to many elements from \( B_t \), an adjustable constraint \( \gamma \in \mathbb{N}_0 \) is introduced to limit the number of possible split/merge components associated with a given element. In the limit case of \( \gamma = 0 \) none of the associated \( F_t \) and \( B_t \) elements are taken and an empty set is associated.

The same procedure is then repeated for the elements in \( B_t \) except 1-to-1 associations are not considered, since these are already covered in the first association scan.

In the current implementation, before the trajectory association problem is formed, it is checked if associations satisfy pairwise association conditions and association constraints that are outlined in the following sections. If not, the violating associations are removed. This is important to reduce the large number of pairwise associations to a set of a manageable size.

While here we consider associations defined between trajectories, in the trajectory graph they are represented by edges between the endpoints of trajectories. Associations are mapped to edges via the following map \( K \):

\[
K : A \rightarrow E = \left\{ \begin{array}{ll}
(\emptyset, T_k) & \rightarrow (\text{Entry}, O_{i,k}) \\
(T_k, \emptyset) & \rightarrow (\text{Exit}, O_{i,k}) \\
(T_k, T_m) & \rightarrow (\text{Translation}, O_{i,k}, O_{i,m}) \\
(T_k, \{T_m\}) & \rightarrow (\text{Split}, O_{i,k}, O_{i,m}) \\
([T_k], T_m) & \rightarrow (\text{Merge}, O_{i,k}, O_{i,m}) \\
\end{array} \right. \quad (9)
\]

where \( k \neq m, m \in \mathbb{N} \).

2.6 Association conditions

- Self-associations are forbidden.
- Time-forward associations only.
- Limited maximum object displacement per frame.
- Limited association range. A primary sphere of influence (SOI) based on the node object’s effective radius

\[
||\vec{r}_0 - \vec{r}_k|| < C \cdot r_{SOI}(S) \quad (10)
\]

where \( C \) is a scale factor for the SOI. A secondary smaller SOI of a fixed size – objects within are always associated.
2.7 Association constraints

Association constraints determine whether or not an association is plausible. Entry and exit associations are always considered plausible. Translation associations are expected to comply with weak mass and momentum conservation laws, which are used to determine if two trajectories are consistent in terms of object motion.

Denote the two trajectory segments within the time window with subscripts 1 and 2 and the connecting edge with subscript $k$. Translation associations are constrained by the maximum linear acceleration $a_c$:

\[
\begin{align*}
2 \cdot \frac{\|\vec{v}_k - \vec{v}_1\|}{\Delta t_k + \Delta t_1} &< a_c \\
2 \cdot \frac{\|\vec{v}_2 - \vec{v}_1\|}{\Delta t_2 + \Delta t_k} &< a_c
\end{align*}
\]  

(11)

where $\vec{v}$ is velocity at the respective trajectory edges and $\Delta t$ is the time difference between the nodes of considered edges.

We also limit the change in direction of movement based on the velocity:

\[
\begin{align*}
\arccos\left(\frac{\vec{v}_k \cdot \vec{v}_1}{\|\vec{v}_k\| \|\vec{v}_1\|}\right) &< (\pi + \epsilon) \cdot \exp\left(-\frac{\|\vec{v}_1\|}{\lambda}\right) \\
\arccos\left(\frac{\vec{v}_2 \cdot \vec{v}_k}{\|\vec{v}_2\| \|\vec{v}_k\|}\right) &< (\pi + \epsilon) \cdot \exp\left(-\frac{\|\vec{v}_k\|}{\lambda}\right)
\end{align*}
\]  

(12)

where $\epsilon$ is an arbitrary small constant, and $\lambda$ controls the maximum tolerable direction deviation with respect to velocity. Higher parameter values mean greater permitted deviations. The direction deviation constraints mimic momentum conservation by expecting objects with greater velocity to be less susceptible to deflection.

Weak mass conservation limits area (e.g., projection area due to transmission contrast imaging) differences between trajectories connected via translation:

\[
T_1, T_2 : \frac{|\langle S_1 \rangle - \langle S_2 \rangle|}{\max(\langle S_1 \rangle, \langle S_2 \rangle)} < \epsilon_t \cdot \frac{\sigma_k}{\langle S_k \rangle}
\]  

(13)

where $S_{1,2}$ are the sets of area measurements, $\sigma_{1,2}$ are the standard deviations for $S_{1,2}$, $k = \{1, 2\}$ and $\epsilon_t$ is the area deviation threshold. Greater thresholds permits higher relative deviations of area.

In the case of split/merge events, weak momentum conservation is unreliable due to surface tension effects, therefore only weak mass conservation is used. This essentially checks if the projection areas of objects before and after splits/merges are consistent:

\[
\frac{|S_0 - \sum_k \langle S_k \rangle|}{\max(S_0, \sum_k \langle S_k \rangle)} < \epsilon_s \cdot \frac{\sigma_k}{\langle S_k \rangle}
\]  

(14)

where $S_k$ and $\sigma_k$, $k \in \mathbb{N}$ correspond to the merge components and/or split products and $\epsilon_s$ is the mass conservation threshold, the lower the value, the more the changes in object areas are expected to comply with mass conservation. Here, a linear relationship is assumed between the mass and the projection area of the bubble. This is a priori wrong, since it leads to an apparent mass defect, although the mass is not physically lost. Up to a certain level, this approach provides satisfactory results.

While bubble sizes physically do change the breakup/coalescence frequency, the tracking performance does not depend on the bubble sizes, but rather on relative area variations for individual bubbles and the diversity of bubble sizes within the field of view (FOV). This is because the breakup/coalescence events are modeled using Equation 14 and translational motion depends on bubble size measurements via Equation 13. Note that in both cases there is normalization with respect to bubble sizes, meaning that Equations 13 and 14 essentially constrain bubble size variations within trajectories and across trajectories considered for split/merge events. The more diverse bubble sizes are and the more consistent the sizes of individual bubbles are in time, the better the tracking performance. Trade-offs are controlled by $\epsilon_s$ and $\epsilon_t$ parameters.

2.8 MHT

The Bayesian formulation of MHT is used Rubio et al. (2012). Due to the offline nature of the algorithm, the framework is greatly simplified and the problem of finding feasible (non-contradictory) sets of trajectories is as follows. Let $A$ be a list of feasible associations and $X$ be a binary vector representing hypothesized trajectory configurations (i.e., sets of association states). The goal is to find the most likely state from $X$ given $A$, that is, to find the maximum a posteriori estimate $X^*$:

\[
X^* = \arg \max_X p(X|A)
\]  

(15)

where due to the Bayes’ theorem one has

\[
p(X|A \propto p(A|X)p(X)
\]  

(16)

where $p(X) = 1$ if the hypothesized associations are not contradictory, meaning that a trajectory is in only one of the hypothesized associations. Given a trajectory configuration, one calculates the likelihood of associations $A_i$ as follows:

\[
p(A_i|X) = \prod_{i=0}^{n} p(A_i|X_i)
\]  

(17)
\[ p(A_i|X_i) = \begin{cases} f(A_i), & X_i = 1 \\ 1 - f(A_i), & X_i = 0 \end{cases} \] (18)

where \( f(A_i) \) is a function that evaluates the likelihood of a single association assumed to be True. Herein \( f(A_i) \) are referred to as statistical functions. A unique statistical function is defined for every type of association.

The trivial way to solve for \( X^* \) is a brute force search verifying every possible \( X \), which is extremely inefficient due to the \( \mathcal{O}(n) = 2^n \) complexity of the search, and is therefore only feasible for a very low number of associations. Reducing the effective \( n \) is an option, but that does not solve the scaling problem which becomes critical for very large measurement sets, i.e., measurements with high number density per frame or very long measurement processes, which is the case in many scientific applications.

Reduction of the effective \( n \) was attempted by representing the set of associations as binary matrix \( (A_y) \) where each row stands for an element of \( F_i \) or \( B_i \) and columns represent the individual associations. Then an undetermined system of linear equations was formed: \( A_yX = b_i \), where \( b_i \) is a vector containing only unity elements. Solving this for the independent variables, they could be then used to solve a problem with a reduced \( n \). While this method yielded true solutions in many cases and reduced complexity, it sometimes failed to yield any solutions and was therefore unreliable.

The proposed approach is to recognize this as an exact cover problem (Knuth 2000), since the solutions of the exact cover problem are by definition sets of associations that yield \( p(X) = 1 \). The problem is reformulated as an exact cover problem as follows:

1. Define the universe as \( U = F_n \cup B_m \)
2. \( A \) is a collection of subsets of \( U \)
3. Solve for collections of elements of \( A \) in compliance with the exact cover problem.

The best known way of solving an exact cover problem is using the Algorithm X (Knuth’s algorithm) (Knuth 2000), which reduces the computational complexity significantly down to consistently sub-exponential. Complexity is reduced further by clustering associations into disjoint sets before formulating and solving the exact cover problem (Brasó and Leal-Taixé 2020).

### 2.8.1 Disjoint sets of associations

Before formulating the exact cover problem, it is possible to further reduce its complexity by separating the associations into disjoint sets.

This is achieved by defining an undirected graph \( (G^*) \) as follows:

A set of disjoint subgraphs is formed from \( G^* \) and then by checking which of the subgraphs each association belongs to, one can form the disjoint sets of associations. This very effectively reduces the size of the universe for the exact cover problem.

#### 2.8.2 Statistical functions

Let \( \mathcal{N}(x, \mu, \sigma) \) be a Gaussian distribution with its mean \( \mu \) and standard deviation \( \sigma \). The propagation probability is measured by:

\[
\begin{align*}
f_1 &= a \cdot \frac{\mathcal{N}(\delta r, 0, \sigma_{dr})}{\mathcal{N}(0, 0, \sigma_{dr})} + (1 - a) \cdot \frac{\mathcal{N}(\delta S, 0, \sigma_{dS})}{\mathcal{N}(0, 0, \sigma_{dS})} \quad (20)
\end{align*}
\]

where \( \delta r \) is the displacement magnitude, \( \delta S \) is the area difference between node objects and \( a \) is the weight adjustment parameter. Both \( \sigma_{dr} \) and \( \sigma_{dS} \) are computed for both \( I_k \) considered for connection via a translation edge.

Entry/exit edge probability is:

\[
\begin{align*}
f_2 &= \frac{1}{1 + \exp(a(y - b))} \quad (21)
\end{align*}
\]

where \( y \) is the vertical coordinate in a 2- or 3-dimensional image and \( a, b \) are control parameters.

The value of \( b \) determines the location on the predetermined axis (in this case vertical) where there is a 0.5 probability for the existence of an Exit/Entry node. It is recommended that \( b \) is set such that this location is near the FOV boundaries where the bubbles are expected to enter/leave.

The magnitude of \( a \) determines how rapidly the probability for the existence of the Exit/Entry nodes changes over the FOV (from 0 to 1 and vice versa). The value should be chosen such that the false positives can be assigned Exit/Entry anywhere in the FOV, but it should be set high enough to avoid always having Exit/Entry as the likeliest solution. The sign of \( a \) determines whether the probability increases or decreases along the axis: \( a > 0 \) implies a decrease from 1 to 0, and vice versa.

The merge/split event probability is also based on weak mass conservation and is computed as follows:

\[
\begin{align*}
f_3 &= \beta \cdot \mathcal{N}(\delta(S_0, S_k), 0, \langle \sigma_S \rangle) \\
&+ (1 - \beta) \cdot \frac{M_k}{M} \cdot \mathcal{N}(\langle \tilde{r}_k \rangle - \tilde{r}_0, 0, \sigma_{dr}) \quad (22)
\end{align*}
\]

where \( \beta \) is the weight adjustment parameter, \( S_0 \) is the area of the split source/merge product, \( S_k \) are the areas of the merge/
split components, \( \vec{r}_0 \) is the position of the split source/merge product, \( \vec{r}_k \) are the positions of the merge/split components, \( M \) and \( M_s \) are the number of involved trajectories and the number of trajectories with available motion prediction (i.e., there are enough points in a trajectory), respectively. Therefore, \( \beta \) determines whether the area or the position consistency is emphasized more for the split/merge components.

2.8.3 Filtering associations

It is not always desirable to immediately add the feasible associations (edges) to the graph since new trajectories might be generated or enter the time window in the next iteration of the time sweep that are better solutions to the problem. Therefore any association considered likely (and the resulting trajectories) must also satisfy tree conditions:

- \( \forall T_k \in F_t : t_k = t \)
- \( \forall T_k \in B_t : t_1 = t + 1 \)
- \( p_k > p_c \)

This is the last step of the time sweep. After this, the set of accepted edges is added to the graph \( G \) via (9).

3 Performance analysis

To assess the performance of MHT-X for scientific applications, we applied it to three cases of bubble flow in liquid metal where object tracking is necessary and offline tracking is appropriate:

- 2D simulations of argon bubble flow in a rectangular vessel.
- Dynamic X-ray radiography of argon bubble flow in a rectangular vessel filled with GaInSn wherein bubbles are injected via a top submerged lance.
- Dynamic neutron radiography of argon bubble chain flow in a rectangular liquid gallium vessel – bubbles are injected at the vessel bottom via a horizontal/vertical tube.

In all three cases segmentation is performed prior to tracing and bubble flow regime is such that bubbles are expected to deform considerably while ascending to the free surface of liquid metal at the top of the vessels.

3.1 2D bubble flow simulation

The first test case is the output of a 2-dimensional simulation of bubbles rising through liquid gallium in a rectangular vessel. The model is a volume of fluid (VOF) simulation of two-dimensional bubble flow in a rectangular vessel with a horizontally directed inlet at the bottom. The numerical model is described in our previous publications (Birjukovs et al. 2020b, a). The flow regime has been adjusted such that bubble trajectories are highly irregular and collisions/splits/merges are frequent. The bubbles have been perfectly segmented in that there are no false positives or detection failures in this case. Bubbles with projection areas below a predefined threshold were not considered. Therefore this is an idealized test of the tracking capabilities of MHT-X in case of a moderate number density of objects of various sizes and variable shapes with frequent interactions. Here, object coordinates and projection area are tracked. Examples of the tracing output are shown in Figs. 3, 4, 5, 6.

Several frames with overlaid reconstructed bubble trajectories are shown in Fig. 3, where trajectories are color coded by bubble ID. The code successfully tracks both large, significantly deforming bubbles, and smaller ones, even in

![Fig. 3 An example of characteristic trajectory patterns and c two splits (purple arrows) in rapid succession resolved in presence of strong deformations and other potentially interfering bubbles. Trajectories are color coded by their IDs](image)
cases of close proximity with colinear motion. Note the two detected bubble splits indicated in Fig. 3b with purple arrows. Another important feature of this case is that bubbles move in a variety of patterns: ascension due to buoyancy, slow oscillatory motion due to entrapment in low velocity zones, downward motion due to a large vortex with counterclockwise mass flow – all of this is representative of realistic flow conditions in two-phase systems.

In Fig. 4 one can see several bubbles tracked across consecutive frames. Note that in Fig. 4(3)-(6) two split events in rapid succession are resolved where bubble shapes before and after breakup are radically different. It is also important to note that MHT-X does not lose track of bubbles despite significant elongation (especially in frames 3-5) and proximity of two more bubbles that, while initially ascending, divert to the left and begin enter almost colinear motion (frames 5-7).

Figure 5 shows two examples of logically connected trajectory sets (families) derived from the established trajectory graph. Note especially Fig. 5a where family members exhibit rather complex trajectories, proximate and even overlapping trajectories. The entire family in Fig. 5a originates...
from a common entry point at the bottom of the field of view (FOV). The graph allows to directly examine the entire trajectory network and qualitatively assess the intensity of collective dynamics and where the interaction events are localized within the FOV.

In addition to visual information regarding bubble motion and interactions, families and the corresponding exported timestamped datasets for further processing (velocimetry, trajectory curvature measurements, shape parameter evolution tracking, etc.), it may also be helpful to visualize the constructed trajectory graph itself—the solution graph for this test case is shown in Fig. 6.

However, even though this test case demonstrates successful tracing for rather complicated flow patterns and bubble interactions, it is somewhat idealized. Bubble projection area conservation is not violated too strongly, i.e., bubbles do not physically vary in volume (the FOV is above the growth region at the inlet), and the segmented dataset is virtually without error or noise. The following two cases address these conditions.

3.2 Dynamic X-ray radiography of bubble flow

The second case stems from a dynamic X-ray imaging of bubble flow at 125 frames per second (FPS) in a rectangular vessel with a top submerged lance (inlet) setup (Akashi et al. 2019) where the bubbles are injected within the FOV. This means that bubble volume, and therefore also the projection area due to X-ray transmission, are generally very different between a bubble that is being ejected from the inlet tip and already detached bubbles. In addition, the bubbles also exhibit substantial deformations including out-of-plane (with respect to the FOV) motion of the argon/GaInSn interface. Moreover, the bubbles are segmented from X-ray images where noise and potentially artefacts are present, therefore bubble shapes are generally not recovered perfectly. Frequent bubble/bubble interactions are to be expected in this flow regime even though their number density is less than in the previous test case.

However, in this case the data regarding local volume fraction over the FOV are available from measurements (Akashi et al. 2019) and bubble volume has been derived and supplied to the tracer. A conservation law of the same form as one for the projection area $S$ in (13), (14), (20) and (22) was added for the tracked bubble volume. The developed algorithm performed well under the above conditions as illustrated in Figs. 7, 8, 9, 10, 11, 12 (the event in Fig. 9 is a special case). As in the previous case, split/merge events proximate in time were resolved (Fig. 7) and the many-to-one and one-to-many events, much more frequent in this case, were also correctly identified (Fig. 8).

A very important special event is shown in Fig. 9, where the code detected a split event. However, it is, in fact, a very rapid, sub-resolution split event followed by a merge event.
From the perspective of the available data, though, this is a many-to-many event that the present version of the code is not yet equipped to deal with. While in this test case only one such event occurs, many cases of bubble flow involve such types of interactions because the temporal resolution of experiments in general may not be sufficient to explicitly separate series of such events with high temporal density. This is especially challenging when some of the tracked properties are not sufficiently strongly conserved. We are currently developing a methodology that will enable MHT-X to treat such events.

Figure 10 illustrates the trajectory graph for this case. Note the more pronounced irregularity in trajectory patterns at the entry node (left) – this has to do with very frequent merge events right after bubble takeoff at the inlet. The topside of the FOV (the right, exit node in Fig. 10) is dominated by split events, but these are much less frequent on average as their onset in the data series is delayed, while coalescence occurs regularly at the bottom of the FOV. Closeups of the entry node region are show in Figs. 11 and 12 where bubble interactions can be seen in greater detail.

As with the 2D case, trajectory families can be visualized, as shown in Fig. 13. The frequent coalescence evident from graphs in Figs. 10, 11, 12 is also clearly seen in Fig. 13a and b especially, where this results in many logically connected trajectories. Note also that some splits occur at the bottom as well, e.g., in Fig. 13a, which occurs
because bubbles sometimes detach from a gas pocket that forms at the bottom of the injection lance rather than directly from the lance nozzle. While this occurs relatively rarely in this test case, some bubbles ascend to the top of the FOV without any interactions whatsoever, e.g., as in Fig. 13d.

For some applications it may be of interest to inspect the split/merge event statistics: event locations by type, their spatial density over the FOV, etc. In addition, retrieving angles at which objects, in this case bubbles, merge or split can yield insights into what drives the observed behavior. In a split event the source bubble produces \( n \) split components. The split angle \( \phi_i \) for the \( i \)-th component is calculated as follows:

1. Using trajectory splines, determine a vector (\( \vec{t} \)) tangent to the trajectory at the moment of the last measurement for the originating bubble.
2. Determine a displacement vector (\( \vec{c}_i \)) between the originating bubble and the \( i \)-th component.
3. Compute \( \phi_i \) via:

\[
\phi_i = \pm \arccos \left( \frac{\vec{t} \cdot \vec{c}_i}{||\vec{t}|| \cdot ||\vec{c}_i||} \right)
\]

The sign convention is tied to that of the unit circle with the zero angle corresponding to the tangent vector direction. Merging events are treated simply as time-reversed split events, therefore the calculations and the sign convention are the same.

Figures 14, 15, 16 are an example of such an analysis. From Fig. 14a and b one can see that the merging events mostly occur at the bottom due to trailing bubbles quickly catching up with leading bubbles, whereas splits occur rather often at the top, where bubbles that are often products of mergers are once again split into smaller bubbles, and at the bottom where bubbles detach (technically, split) from the gas pocket that consistently forms at the inlet nozzle.

Figure 15 shows the statistics for merging (a) and splitting (b) angles. Note that merges mostly occur about zero degree angles – this makes sense given mostly vertical collisions between leading and trailing bubbles near and above the inlet nozzle. Splits, however, exhibit a much less ordered distribution with no consistently clear peaks, but there is
a pronounced asymmetry in that negative split angles have greater magnitudes.

Figure 16 offers further insight indicating that large angle splits are concentrated in the upper (Fig. 16b) left (Fig. 16a) area of the FOV and are largely responsible for the upper density maximum zone in Fig. 14b. It is also revealed that high angle merges are distributed mostly in the bottom 2/3 of the FOV.

In this test case the bubble dynamics are a bit more complicated to resolve for MHT-X despite the lower number density of objects compared to the first test case. Nevertheless, the data set is relatively clean because the signal-to-noise ratio (SNR) in the original X-ray transmission images is quite high and false positives are not present. To demonstrate the code’s robustness against noisy input and false positives, we present the results from the third test case in the following section.

3.3 Dynamic neutron radiography of bubble flow

The third case uses the data obtained by means of dynamic neutron imaging of argon bubble flow in liquid gallium at 100 FPS in a system described in detail in Birjukovs et al. (2020b), Birjukovs et al. (2020a), Birjukovs et al. (2021). The characteristic feature of neutron radiography images acquired with a high frame rate for thick liquid metal vessels is the very low image SNR. Because of this, even with advanced noise filtering and segmentation, the data provided as input to MHT-X is inevitably noisy in that bubble centroid position uncertainties are considerable and much greater.

Fig. 14  a Locations of split (red) and merge (gray) events within the FOV and maps of normalized (b) merge and (c) split event density. b and c are normalized separately and share the color legend (right). X and Y coordinates are in pixels. The red-dashed circles are inlet locations. Density maps are computed using a Gaussian kernel over the count area density function with Silverman’s bandwidth estimation.

Fig. 15  Direction angle histograms for a merge components and b split components.
than in the first two test cases shown herein. In addition, occasional false positives and detection failures may occur, further complicating tracing. To isolate these effects, image sequences with no bubble interactions were chosen, i.e., the average bubble spacing is sufficient to avoid collisions. The flow regime expected for this experimental setup (Birjukovs et al. 2020b, a, 2021) and the moderate gas flow rates considered is the bubble chain, where the distance between bubbles is determined by the respective value of the gas flow rate (low flow rates produce quasi single-bubble flow). Figures 17 and 18 illustrate characteristic bubble trajectories reconstructed with MHT-X. Here (a) and (b) show zig-zagging bubbles and (c) and (d) show mostly rectilinear trajectories due to flow stabilization with applied magnetic field.

**Fig. 16** Merge and split angle distributions over a X and b Y coordinates (in pixels) in the FOV

**Fig. 17** Neutron radiography images with highlighted trajectories color coded by IDs, bubbles (white-dashed circles) and false positives (red-dashed circles). Up to a number of latest trajectory edges are shown in a–d for visual clarity
One can clearly see that bubbles (white-dashed circles in Fig. 17) are largely shrouded by image noise. Despite this and the resulting noise in the object dataset, it is seen that the algorithm performs well and long, consistent trajectories spanning the entire FOV are recovered. Note also that in Figure 17d there are two false positives (red dashed circles) that were overlapping in time with true detections—these were resolved as isolated nodes (1-node trajectories) in that within the solution graph they are only connected to the entry and exit nodes. In Fig. 18b, there is an edge with a lower probability in the upper part of the trajectory – this is an example of MHT-X correctly extrapolating and connecting trajectory fragments across a frame with a detection failure event. Figure 18 also shows that the code is indeed resilient to noisy data, which is particularly evident in cases (c) and (d). The solution graph, in turn, is shown in Fig. 19 where one can see how radically it differs from Fig. 10 due to the absence of bubble interaction events. It must be noted that the graphs shown in this paper are not representative of data noise because of the way graphs are transformed in GePhi.

4 Computational performance

Since the performance of the entire algorithm mostly depends on that of the Algorith-X, a benchmark is run to test it for the association types that may occur in tracing applications with split/merge events. The benchmark is performed as follows:

1. Generate 2 groups of \( m \) and \( n \) objects.
2. Association condition fulfillment is set as probabilistic with a likelihood of 0.3.
3. Association constraint fulfillment is set as probabilistic with a likelihood of 0.5.
4. Separate the set of associations into disjoint sets.
5. Log the amount of associations and how much computational time it takes for the Algorithm-X to exhaust the search-space for each disjoint set.

This was repeated 10000 times for all combinations of \( 6 \leq m, n \leq 10 \) with \( \gamma = 3 \) considering only unique unordered \((m, n)\) pairs. The resulting ~ 250K datapoints were separated
uniformly into bins by the association count with bin with equal to 1. The resulting bins were then also binned vertically into 100 equally spaced bins from minimum to maximum execution time. Then a discrete normalized density distribution was generated. The resulting computation time distributions versus association count within a disjoint set are shown in Fig. 20.

One can see in Fig. 20a that the worst case scenario is exponential as it should be. The exponential time cases, however, are encountered very rarely and most of the cases are solved significantly faster.

It is worth noting that increasing the dimensionality of the problem does not improve or worsen the performance of the algorithm. The limiting factors are object number density and collision frequency, as well as the mean number of objects in split/merge events. Extra spatial dimensions simply add more coordinates to track.

5 Further extensions and improvements

While the current version of the algorithm is already very versatile, as indicated by the above results, there are still potentially many ways it can be improved. We aim to make it more broadly applicable and enable solving tracing problems with greater object density and more adverse conditions as seen in the cases outlined in the Introduction. The core components—graph architecture, Bayesian MHT, Algorithm X—must be backed by appropriate lower-level methods. To this end, it is planned to do the following in the future:

- Replace the spline-based extrapolation with a Kalman filter predictor.
- Develop a treatment for many-to-many object association events.
- Implement a feedback loop that will enable coupling with image processing pipelines for iterative reinforced object detection and tracking.
- Implement object cluster detection and joint tracking for improved tracing accuracy when dealing with swarms of coherently moving objects.
- Introduce a scheme for edge probability re-evaluation (currently static once assigned) during successive graph sweeps to enable removing/reinforcing edges as more context becomes available after each time window sweep iteration.
- Optimize performance: use a C-compiled graph computation package instead of the currently used Python library; parallelize trajectory construction, association evaluation and visualization routines.

There are potentially more modifications that currently are not considered. In addition, appropriate heuristics (statistical functions, etc.) for tracking of particles and other objects will be developed as necessary.

6 Conclusions and outlook

We have developed and demonstrated the capabilities of an offline Bayesian multiple hypothesis tracker with a directed graph architecture that uses Algorithm X to solve the optimal association problem as an exact cover problem. The showcased test cases indicate that the current implementation is robust enough to process cases with relatively high object number density, in presence of data noise, false positives and detection failures. The algorithm is capable of resolving one-to-many split and many-to-one merge events for objects with variable shapes and parameters.

Fig. 20 Computational performance of the Algorithm-X for associations feasible in tracing problems with split/merge events. a The entire set of benchmark data points and b a subset of the benchmark data with execution time between 10 ms and 8 s which is roughly 15% of the entire data set
In its current state and especially as the outlined improvements are implemented, we expect that the code will find use in many areas of research. Beyond that the code is currently in use for the development of a dynamic mode decomposition code for the analysis of output of bubble flow simulations, as well as for bubble shape analysis, including shape evolution tracking and phase boundary velocimetry.

Acknowledgements This research is a part of the ERDF project “Development of numerical modelling approaches to study complex multiphysical interactions in electromagnetic liquid metal technologies” (No. 1.1.1.1/18/A/108). Neutron images were acquired at the Swiss spallation neutron source SINQ, Paul Scherrer Institute (PSI). The authors are grateful to Pavel Tritic (PSI), Jevgenijs Telicic (UL), Jan Hovind (PSI) and Knud Thomsen (PSI) for invaluable support in the neutron radiography experiments, and express gratitude to Natalia Shevchenko (HZDR) for her key role in the X-ray radiography experiments. The authors would like to acknowledge the math.stackexchange.com user Watercrystal for pointing out that the association optimization problem can be treated as an exact cover problem. Her involvement, while in the form of a single comment, was crucial to the development of this algorithm: https://math.stackexchange.com/questions/3720630/finding-the-solution-set-for-binary-matrix-and-vector-multiplication/3720965#3720965.

Author Contributions Peteris Zvejnieks is the main developer of MHT-X and has written all parts of the Python implementation. Both Peteris Zvejnieks and Mihails Birjukovs contributed to algorithm and method development. Martins Klevs performed the bubble flow simulations and provided the data for analysis. Megumi Akashi processed the X-ray radiography images and provided the data for analysis. Neutron radiography images were processed by Mihails Birjukovs. Visualization was done by Peteris Zvejnieks and Mihails Birjukovs, both of whom also came up with the original manuscript draft. Sven Eckert and Andris Jakovics were responsible for funding acquisition and research supervision. All co-authors contributed to manuscript editing and review prior to submission.

Data availability Both input for and output of MHT-X, as well as associated visuals are available on demand – please contact the corresponding authors.

Code availability The MHT-X code is open source and is currently available on GitHub: https://github.com/Peteris-Zvejnieks/MHT-X. It is frequently updated and a comprehensive documentation is also currently in the works.

References

Akashi M, Keplinger O, Shevchenko N, Anders S, Reuter M (2019) X-ray radioscopic visualization of bubbly flows injected through a top submerged lance into a liquid metal. Metall Mater Trans B. https://doi.org/10.1007/s11663-019-01720-y

Anders S, Noto D, Seilimayer M, Eckert S (2019) Spectral random masking: a novel dynamic masking technique for piv in multiphase flows. Exp Fluids 60:68. https://doi.org/10.1007/s00348-019-2703-8

Anders S, Noto D, Tasaka Y, Eckert S (2020) Simultaneous optical measurement of temperature and velocity fields in solidifying liquids. Exp Fluids. https://doi.org/10.1007/s00348-020-2939-3

Birjukovs M, Dzelme V, Jakovics A, Thomsen K, Tritic P (2020) Phase boundary dynamics of bubble flow in a thick liquid metal layer under an applied magnetic field. Phys Rev Fluids. https://doi.org/10.1103/PhysRevFluids.5.061601

Birjukovs M, Dzelme V, Jakovics A, Thomsen K, Tritic P (2020) Argon bubble flow in liquid gallium in external magnetic field. Int J Appl Electromagn Mech 63:1–7. https://doi.org/10.3233/JAE-209116

Birjukovs M, Tritic P, Kaestner A, Hovind J, Klevs M, Gawryluk DJ, Thomsen K, Jakovics A (2021) Resolving gas bubbles ascending in liquid metal from low-snr neutron radiography images. Appl Sci. https://doi.org/10.3390/app11209710

Blackman S (2004) Multiple hypothesis tracking for multiple target tracking. IEEE Aerosp Electron Syst Magaz 19:5–18. https://doi.org/10.1109/MAES.2004.1263228

Brasó G, Leal-Taixé L (2020) Learning a neural solver for multiple object tracking. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6246–6256. https://doi.org/10.1109/CVPR42600.2020.00628

Cox JJ, Hingorani SL (1996) An efficient implementation of Reid’s multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. IEEE Trans Pattern Anal Mach Intell 18:138–150. https://doi.org/10.1109/34.481539

Dzelme V, Jakovics A, Vencels J, Köppen D, Baake E (2018) Numerical and experimental study of liquid metal stirring by rotating permanent magnets. In: IOP Conference Series: Materials Science and Engineering 424:012047. https://doi.org/10.1088/1757-899X/424/1/012047

Heitkam S, Lappan T, Tritic P, Eckert K (2019) Tracking of particles in froth using neutron imaging. Chemie Ingenieur Technik. https://doi.org/10.1002/cite.201800127

Keplinger O, Shevchenko N, Eckert S (2018) Visualization of bubble coalescence in bubble chains rising in a liquid metal. Int J Multiph Flow 105:159–169. https://doi.org/10.1016/j.ijmultiphaseflow.2018.04.001

Keplinger O, Shevchenko N, Eckert S (2019) Experimental investigation of bubble breakup in bubble chains rising in a liquid metal. Int J Multiph Flow 116:39–50. https://doi.org/10.1016/j.ijmultiphaseflow.2019.03.027

Kim C, Li F, Ciptadi A, Rehg J (2015) Multiple hypothesis tracking revisited, pp 4696–4704. https://doi.org/10.1109/ICCV.2015.553

Knuth DE (2000) Dancing links arXiv:cs/0011047. Accessed 2020-12-06

Lappan T, Sarma M, Heitkam S, Tritic P, Mannes D, Eckert K, Eckert S (2020) Neutron radiography of particle-laden liquid metal flow driven by an electromagnetic induction pump. Magneto-hydrodynamics 56:167–176

Obiso D, Akashi M, Kriebitzsch S, Meyer B, Reuter M, Richter A (2020) Cfd modeling and experimental validation of top-submerged-lance gas injection in liquid metal. Metall Mater Trans B. https://doi.org/10.1007/s11663-020-01864-2

Reid D (1979) An algorithm for tracking multiple targets. IEEE Trans Autom Control 24(6):843–854. https://doi.org/10.1109/TAC.1979.1102177

Rubio JC, Serrat J, López A (2012) Multiple target tracking and identity linking under split, merge and occlusion of targets and observations. In: ICPRAM 2012—Proceedings of the 1st international conference on pattern recognition applications and methods 2, 15–24. https://doi.org/10.5220/0003710600150024

Sarma M, Ščepanskis M, Jakovics A, Thomsen K, Nikoluškins R, Vontobel P, Beinerts T, Bojarevics A, Platacias E (2015) Neutron radiography visualization of solid particles in stirring liquid metal. Phys Proc 69:457–463. https://doi.org/10.1016/j.phpro.2015.07.064
Ščepanskis M, Sarma M, Vontobel P, Trtík P, Thomsen K, Jakovics A, Beinerts T (2017) Assessment of electromagnetic stirrer agitated liquid metal flows by dynamic neutron radiography. Metall Mater Trans B 48:1045–1054. https://doi.org/10.1007/s11663-016-0902-8

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.