Stereo Reconstruction of Droplet Flight Trajectories

Luis A. Zarrabeitia, Faisal Z. Qureshi *Member, IEEE,* Dhavide A. Aruliah

Abstract—We developed a new method for extracting 3D flight trajectories of droplets using high-speed stereo capture. We noticed that traditional multi-camera tracking techniques fare poorly on our problem, in part due to the fact that all droplets have very similar shapes, sizes and appearances. Our method uses local motion models to track individual droplets in each frame. 2D tracks are used to learn a global, non-linear motion model, which in turn can be used to estimate the 3D locations of individual droplets even when these are not visible in any camera. We have evaluated the proposed method on both synthetic and real data and our method is able to reconstruct 3D flight trajectories of hundreds of droplets. The proposed technique solves for both the 3D trajectory of a droplet and its motion model concomitantly, and we have found it to be superior to 3D reconstruction via triangulation. Furthermore, the learned global motion model allows us to relax the simultaneity assumptions of stereo camera systems. Our results suggest that, even when full stereo information is available, our unsynchronized reconstruction using the global motion model can significantly improve the 3D estimation accuracy.

Index Terms—Stereo reconstruction multi-target tracking multi-view geometry nonlinear motion parameter estimation

1 Introduction

We present an end-to-end computer vision system for automated tracking and extraction of individual three-dimensional (3D) flight trajectories of liquid droplets captured using multiple high-speed video cameras (i.e., 1300 frames per second at 1280×800 resolution). This system enables automated analysis of videos of *bloodletting events*. In each blooedleting event, a riot ball or large pellet is fired from a paintball gun into a synthetic gel filled with *porcine* or transfer blood (to mimic human flesh). Our system comprises monocular and stereo droplet tracking routines that process videos captured and extract 3D trajectories of individual droplets. The system described herein is unique—to the best of our knowledge—as it is flexible with respect to choice of differential equation-based dynamics and it is capable of tracking many (on the order of a hundred) individual indistinguishable droplets and reconstructing 3D trajectories with minimal user interaction.

Videos of the bloodlettings are captured to enable validation of computational models of droplet flight for *Bloodstain Pattern Analysis (BPA)* [1]. BPA comprises a body of techniques employed by forensic specialists to infer properties of violent bloodletting events—e.g., location of impact, type of weapon used, etc.—from bloodstains found at crime scenes. There is considerable interest in moving away from stringing—a geometric reconstruction technique in BPA based on the assumption that droplets travel in straight lines—in favor of models with more realistic physical assumptions. For instance, Cechetto et. al. [2] consider probabilistic inverse dynamics incorporating gravity and a linear model of aerodynamic drag (stringing implicitly requires that effects of gravity and aerodynamic drag are negligible). Our work complements that in [2] and [1] in providing a computational framework by which putative model dynamics can be contrasted in their ability to characterize motion captured in actual experiments. In addition, the optimization-based extraction of 3D trajectories from videos permits the construction of an empirical prior distribution of model parameters like drag coefficients or droplet speeds (estimation of such quantities from crime scene photographs is practically impossible).

There are three principal contributions in the present work. Firstly, the 2D tracking procedure in Sec. 3 can disambiguate hundreds of virtually identical droplets in the presence of noise and inter-droplet occlusions of moderate duration. Moreover, our model-based reconstruction system can be used for reconstruction of 3D droplet trajectories without triangulation (see Sec. 4). Secondly, our framework is flexible and general with respect to synchronization and ODE-based dynamics. That is, it is possible to relax the requirement of synchronized cameras for 3D trajectory reconstruction as described in Sec. 5. Our approach works within a broad class of object dynamics described by linear or non-linear ordinary differential equation (ODE) models. Lastly, when the system is used with actual video data, the principal assumption underlying stringing in BPA is demonstrated to be false—namely that a droplet travels in a straight line for practical purposes, i.e., that the effects of gravity and aerodynamic drag can safely be ignored over the time and length scales relevant in BPA. Indeed, the results in Sec. 7 suggest that droplet trajectories are not parabolic.

2 Literature Review

The object-tracking problem consists of successfully identifying objects as they move through a scene recorded by a set of sensors. A broad survey of tracking algorithms can be found in [3]. In the present work, we focus on optical tracking of multiple targets where the sensors—cameras—produce sequences of video frames that contain images of the moving objects. We consider the cameras to be stationary and of fixed orientation.

Tracking flight paths of liquid droplets across a large number of video frames presents considerable challenges. In particular, tracking can be formulated by establishing

• *Faculty of Science, University of Ontario Institute of Technology, Oshawa, ON, L1H 7K4, Canada.*
E-mail: {luis.zarrabeitia,faisal.qureshi,dhavide.aruliah}@uoit.ca

TABLE 1: Summary of the notation used in Sections 3 to 7.

2D tracking	
N	Number of frames.
k, t_k	A frame number and its corresponding timestamp.
$\mathcal{B}^{(k)}$	Sensor measurements (blobs) in frame k.
$\mathcal{T}^{(1:k)}$	Set of trajectories computed up to frame k.
$\tau \in \mathcal{T}^{(k)}$	A single path.
$b^{(k)} \in \mathcal{B}^{(k)}$	Any single blob of $\mathcal{B}^{(k)}$.
$\mathcal{M}$	Set of constraints for the matching algorithm.
$\mathcal{E}$	Set of allowable edges for the matching algorithm.
3D reconstruction	
$\{\zeta_1, \zeta_2\}$	Set of cameras.
M	Set of measurements of a single target, comprised of tuples m_i of:
t_i	timestamp of the i-th measurement,
$\mathbf{x}_i$	coordinates recorded by the i-th measurement, and
ς_i	camera that recorded the i-th measurement. The same camera $\varsigma \in \{\zeta_1, \zeta_2\}$ may record several measurements of the same target, in which case, $\varsigma_i = \varsigma$ for all recordings of the same camera ς.
φ_ς	Projection operator corresponding to camera ς. When referring to a single trajectory, φ_{ς_i} is abbreviated as φ_i.
$\mathbf{P}_\varsigma$	Projection matrix corresponding to camera ς. When referring to a single trajectory, $\mathbf{P}_{\varsigma_i}$ is abbreviated as $\mathbf{P}_i$.
$\boldsymbol{\lambda}$	True, unknown $3D$ location of the object.
$\mathcal{L}_{\boldsymbol{\alpha}}(t)$	Motion model for the target (family of functions depending on the parameter $\boldsymbol{\alpha}$).
τ_i^*	Trajectory τ recorded by camera ς_i, augmented with the timestamps of each measurement.

correspondences—matchings in a bipartite graph—between targets—2D projections of 3D objects—in successive video frames. Liquid droplets are semi-transparent and quasi-deformable as well as being of very similar appearance and size—their projections are typically only a few pixels across. As such, tracking techniques that rely upon detection of feature points (see, e.g., [3]) cannot easily distinguish and identify individual droplets. Furthermore, establishing correspondences between targets across a sequence of frames is hindered by the presence of droplet occlusions, misdetections, deformations, or even droplets falling in or out of the field of view. We adopt an approach similar to that followed in [4] to track individual targets in monocular high-speed videos. However, our typical scenario involves significantly more targets than in [4] and, at the same time, has no feature points or similar information *a priori* to assist construction of matchings between successive frames.

Many multi-target tracking schemes have been developed recently (particularly in the context of surveillance). Most of these algorithms combine a model estimation strategy with a data association strategy. Because human or animal motion is less predictable than a droplet's ballistic motion, typical person-tracking algorithms apply probabilistic or Markovian motion models of target motion. For instance, [5] models human motion using Markov-Chain Monte-Carlo Data Association; see also [6] and [7] for related non-deterministic approaches. Zamir, et. al. discuss a more deterministic method in [8], interpreting the tracking problem as a clique-finding problem.

Frame-by-frame tracking algorithms solve *assignment problems* [9] in every frame; i.e., a matching between objects (targets) must be found between each pair of successive video frames. The Kuhn-Munkres Algorithm (also known as the Hungarian algorithm [10], [11]) was proposed in the mid-1950s to solve linear assignment problems in polynomial time. Bourgeois and Lassalle extended the Kuhn-Munkres algorithm to handle rectangular problems in 1971 [12]. A more recent effort, in addressing the *sailor assignment problem*, produced a variant of the rectangular method for sparse graphs and multi-objective problems [13]. For more on assignment problems and multi-target tracking, see [9], [14], and [15].

Multi-Hypothesis Tracking (MHT) [16] and Greedy Optimal Assignment [17] are two algorithms for multi-target tracking. The latter algorithm was used to track a large number of bats in [18]. A similar strategy was applied in [4] to track live insects. Another approach for insect-tracking uses a *particle filter* [19], [20]; both of these studies involve a Markov Random Field to model insect interactions and the assumption that targets actively avoid collisions. These attempts were extended in [21] to account for split and merged measurements. Investigations of a related multiple insect-tracking problem—flies in this case—employ a multi-camera system to generate 3D trajectories of targets in flight (see [22], [23], [24]). Tracking flying insects is closer to our tracking problem than those motivated by surveillance or tracking crawling insects. Insect flight paths are arguably harder to predict than ballistic trajectories of droplets. On the other hand, the fly-tracking problem is simpler than ours in some respects: our problem has many more targets moving at high speeds that may split or merge in flight; as such, our droplet-tracking problem is much more sensitive to prediction errors.

Recovering the three-dimensional structure of a scene (trajectories of individual droplets in our scenarios) requires matching features harvested from two or more images captured by distinct, spatially-separated, calibrated cameras. Once features in distinct camera images are matched, reconstruction proceeds via triangulation. We refer the reader to [25] for an accessible exposition on multiple-view geometry, camera calibration, triangulation, and estimation of scene depth.

In [26], [27], Tyagi et. al. present a unified approach to object tracking in 3D. Visual features are extracted from each view and are used to construct 3D features, which are then tracked. A similar strategy is used in [28]. These methods require simultaneous views from at least two cameras, in addition to a robust scheme to associate targets in different views. Kausler, et. al. also developed a method [29] to track cells directly in 3D. Their method, however, uses 3D images rather than stereo optical sensors. Our droplet tracking scenario does not satisfy these requirements.

Recently, Park et. al. considered in [30] the problem of estimating scene depth when full stereo information is not available. Their strategy extracts motion parameters from images captured from different locations and times. An important assumption in this case is that the motion of the tracked object can be approximated by a linear combination of basis functions specified *a priori*. We extend and exploit a different family of motion models to improve estimation of 3D droplet trajectories

in Sec. 5.2 and in [31]; our formulation allows us to relax the assumption that the stereo cameras need to be synchronized.

3 Single Camera Tracking

We can describe multi-target tracking in monocular videos as an association problem, where targets visible in each frame are associated with targets that appear in subsequent frames. We now explain our strategy for tracking multiple droplets using a single camera setup. The ability to track multiple droplets in monocular videos is key to the overall strategy for reconstructing 3D trajectories presented later in this paper.

Our method begins by identifying foreground blobs—connected sets of pixels—in video frames captured by a single camera. The details of the background subtraction process are available in [32]. The blobs corresponding to true droplets can be small, with a diameter of 5 to 10 pixels. It is therefore hard to distinguish *a priori* which blobs are misdetections and which ones correspond to actual droplets. To maximize the number of real droplets captured we defer blob pruning to a later stage, i.e., we developed a track-before-detect algorithm to track the real droplets and discard the false positives.

3.1 Problem Description

Let $\mathcal{B}^{(k)}$ denote the set of blobs in frame k (for a summary of the notation used in this and the following sections, see table 1). For our purposes, a blob $b^{(k)} \in \mathcal{B}^{(k)}$ is a connected component of the foreground pixels in frame k. Some of these blobs are the result of misclassifications, when a background pixel is erroneously detected as foreground. We refer to these false positives as "noise." Conversely, it is possible for some droplets to escape detection. A trivial scenario occurs when a droplet leaves the field of view of the camera. More commonly, the background segmentation process may fail to detect a droplet that is too small or too similar to the background. An uncommon, but possible scenario occurs when a droplet is partially occluded by another, causing both droplets to register as a single blob. Therefore, we cannot regard $\mathcal{B}^{(k)}$ as the true set of targets to track, but merely as our best available approximation.

The tracking problem consists of estimating the set of trajectories $\mathcal{T}^{(1:k)}$ of the targets given the blob sets $\mathcal{B}^{(1)}, \cdots, \mathcal{B}^{(k)}$ harvested from frames $1, \ldots, k$. Each single trajectory $\tau \in \mathcal{T}^{(1:k)}$ is comprised of a sequence of blobs $\langle b^{(i_1)}, \cdots, b^{(i_s)} \rangle$, $1 \leq i_1 < \cdots < i_s \leq k$ harvested from $\mathcal{B}^{(i_1)}, \cdots, \mathcal{B}^{(i_s)}$, respectively. Observe that, because targets are not necessarily detected in every frame, a trajectory τ does not, in general, contain one blob per frame.

Frame-by-frame tracking algorithms are based on recursively assigning the blobs extracted from each new frame $\mathcal{B}^{(k+1)}$ to the set of partial trajectories $\mathcal{T}^{(1:k)}$ to obtain the new $\mathcal{T}^{(1:k+1)}$. Many single-hypothesis tracking algorithms, such as [4], link the targets in $\mathcal{B}^{(k)}$ directly to those in $\mathcal{B}^{(k+1)}$. Such approaches work well when the cardinality of the set $\mathcal{B}^{(k)}$ does not vary significantly between frames k and $k+1$. For the droplet tracking problem, this assumption can break down in a number of ways:

1) a false target is spuriously detected in frame k, i.e., pixels contaminated by noise are mistaken for a legitimate physical object in the scene;
2) a target seen in frame k is not detected in frame $k+1$;
3) a true target is temporarily occluded by another object in frame $k+1$;
4) a target temporarily leaves the field of view between frames k and $k+1$;
5) a target returns to the field of view between frames k and $k+1$;
6) a target permanently disappears or leaves the field of view between frames k and $k+1$; and
7) a target is first detected between frames k and $k+1$.

Our tracking routine must be able to handle these situations, particularly the most common (1, 2, 6 and 7). The failure scenario 3 appears to be uncommon. Together with 2 and 4, they are the false negative scenarios.

In contrast, a trajectory may stop permanently (condition 6) when it goes beyond the side or bottom edges of the frame, but also when it hits the ground or a wall. Similarly, trajectories do not necessarily start (condition 7) in the first frames and around the impact area, as droplets are continuously produced by the falling container throughout the recording, and long-lost droplets may be detected again. Therefore, unlike other tracking problems, such as in [33], we cannot assume that trajectories start and end in predetermined regions of the frame.

3.2 Multi-Target Tracking as an Assignment Problem

As suggested earlier, multi-target tracking can be seen as solving a series of Assignment Problems (APs). Assignment or matching problems constitute a fundamental class of problems in combinatorial optimization [9]. Specifically, consider disjoint finite sets of vertices A and B and let $\text{cost} : A \times B \to \mathbb{R}$ be a cost function associated with putative edges connecting vertices from A and B. Moreover, let $\mathcal{M} \subset \mathcal{P}(A \times B)$ be a prescribed set of allowable matchings between A and B. The assignment problem associated with this cost function is an optimization problem:

$$\underset{M \in \mathcal{M}}{\operatorname{argmin}} \sum_{(a,b) \in M} \text{cost}(a,b). \tag{1}$$

That is, the goal of an assignment problem is to construct a matching associating vertices from A with vertices in B such that the total cost is minimized.

If $|A| = |B|$ and $\mathcal{M}$ consists of all possible matchings that cover both A and B, the assignment problem is called a *Linear Assignment Problem* (LAP). If $|A| \neq |B|$, and $\mathcal{M}$ consists of all matchings that cover the smaller set, the assignment problem is called a *Rectangular Linear Assignment Problem* (RLAP). Observe that the constraint $\mathcal{M}$ on the set of possible matchings is necessary; otherwise, for many cases, the minimizer will simply be the empty matching $M = \emptyset$.

Assuming for the moment that the constraint $\mathcal{M} \subset \mathcal{P}(A \times B)$ is known, we shall assume henceforth that

$$\text{“}M = \text{SolveAP}(A, B, \text{cost}, \mathcal{M})\text{”} \tag{2}$$

means "Solve (AP) by computing the matching $M \in \mathcal{M}$ that minimizes the sum in (1)." We implicitly assume that a solution of AP exists as does a reasonable algorithm for computing it. We use the techniques in [13] to compute M in (2). There are many situations when some of the elements in A are not connected to any elements in B. Such assignment problems can be solved using a variation of Kuhn-Munkres algorithm, which assigns infinite cost to the missing edges (see [13] for details).

Balch et. al.[4], for example, treat multi-target tracking as an assignment problem. Specifically it sets up the following assignment problem between every two successive frames, k and $k+1$:

$$M^{(k)} = \text{SolveAP}(\mathcal{B}^{(k)}, \mathcal{B}^{(k+1)}, \text{dist}, \mathcal{M}), \tag{3}$$

where $\mathcal{M}$ consists of all matchings that cover either $\mathcal{B}^{(k)}$ or $\mathcal{B}^{(k+1)}$ and

$$\text{dist}\left(b^{(k)}, b^{(k+1)}\right) = \left\| b^{(k)} - b^{(k+1)} \right\|_2. \tag{4}$$

Here the cost of an edge is the Euclidean distance between the centroids of the two blobs that define that edge. The partial paths $\mathcal{T}^{(1:k)}$ are formed by successively concatenating the matches from frames 1 to k. This constraint set $\mathcal{M}$ is suitable for cases 6 and 7 listed in Sec. 3.1; however, it cannot deal with situations when a blob temporarily disappears.

In this work instead, we set up blob tracking as an assignment problem as follows:

- "A" is a set of partial trajectories $\mathcal{T}^{(1:k)}$; and
- "B" are the blobs $\mathcal{B}^{(k+1)}$ detected in the next frame.

Starting with $\mathcal{T}^{(1:1)} = \left\{ \langle b \rangle | b \in \mathcal{B}^{(1)} \right\}$, we construct $\mathcal{T}^{(1:k+1)}$ recursively by augmenting the paths in $\mathcal{T}^{(1:k)}$ with the solution of the assignment problem and adding the new trajectories that may have been started in $\mathcal{B}^{(k+1)}$. The desired matching at each step is given by

$$M^{(k)} = \text{SolveAP}(\mathcal{T}^{(1:k)}, \mathcal{B}^{(k+1)}, \text{dist}, \mathcal{M}), \tag{5}$$

where dist is defined using a path-based predictive model described in Section 3.3.

Observe that this formulation does not necessarily require that blobs in a given path are detected in successive video frames (i.e., that $b^{(k)} \in \mathcal{B}^{(k)}$ absolutely must be followed by some $b^{(k+1)} \in \mathcal{B}^{(k+1)}$). To deal gracefully with the situations enumerated in Sec. 3.1, the constraint set $\mathcal{M}$ should be chosen so that partial paths in $\mathcal{T}^{(1:k)}$ may not be matched to blobs in $\mathcal{B}^{(k+1)}$ if there are no *suitable* unmatched blobs in $\mathcal{B}^{(k+1)}$. For a distance metric $\text{dist} : \mathcal{T}^{(1:k)} \times \mathcal{B}^{(k+1)} \to \mathbb{R}$ and a predefined *tracking threshold* $\varepsilon_{\text{track}}$, the constraint set $\mathcal{M}$ consisting of all the matches M that

1) $\forall \langle \tau, b^{(k+1)} \rangle \in M$; $\text{dist}(\tau, b^{(k+1)}) \leq \varepsilon_{\text{track}}$; and
2) if $\text{dist}(\tau, b^{(k+1)}) \leq \varepsilon_{\text{track}}$, then M covers at least τ or $b^{(k+1)}$, but not necessarily both,

has the desired properties.

In practice, it is infeasible to enumerate the constraint set $\mathcal{M}$ explicitly. Instead, customized matching algorithms are developed for specific classes of constraints. For instance, the sparse variant of Kuhn-Munkres discussed in [13] solves assignment problems where not all edges are present or allowable (e.g., condition 1 above) and maximal coverage is desired (condition 2). We rewrite (5) as

$$M^{(k)} = \text{SolveSparseAP}(\mathcal{T}^{(1:k)}, \mathcal{B}^{(k+1)}, \text{dist}, \mathcal{E}), \quad \mathcal{E} \subseteq \mathcal{T}^{(1:k)} \times \mathcal{B}^{(k+1)}, \tag{6}$$

where the sparsity constraint $\mathcal{E}$ is the set of edges allowed in the matching.

3.3 A Deterministic Fixed-Tail Quadratic Filter

We now define the distance function that is used to assign the edge cost between a partial path $\tau \in \mathcal{T}^{(1:k)}$ and a blob $b^{(k+1)} \in \mathcal{B}^{(k+1)}$ detected in frame $k+1$. The basic idea is to use the partial paths up to and including frame k to predict the location of the blobs in frame $k+1$. The edge costs are then defined simply as the distances between these predicted locations and the blobs detected in frame $k+1$. Motion models constructed using partial paths are used to predict the location of blobs in the $(k+1)$th frame.

During our experiments we observed that the paths traced by droplets in the image plane are locally parabolic. Although a parabolic model does a poor job of predicting the motion over long duration, it still can be used for short term predictions. Bearing in mind that we are dealing with high-speed video footage, a parabolic model is accurate for a small number of frames, sufficient to deal with cases of misdetections. Consequently, we model droplet motions using quadratic polynomials, using measurements from a fixed number of past frames to predict the locations in future frames. The idea of using local approximations for the motion also appear in [33], using B-splines to model the local motion of pedestrians.

Given a partial path $\tau \in \mathcal{T}^{(1:k)}$ corresponding to some droplet's motion up to and including frame k, we use the slice notation $\tau^{(i:j)}$ to denote the maximal subsequence of τ containing only blobs from frames $i \geq 1$ to $j \leq k$. We define $q(k;\tau) : \mathbb{Z}^+ \to \mathbb{R}^2$ as the least squares polynomial of degree at least 2 that fits the data in the sequence of blobs τ. Then, the estimated target position in frame $k+1$ is given by $q(k+1;\tau^{(k-l:k)})$

We now have the machinery to define a cost function for (5):

$$\text{dist}\left(\tau, b^{(k+1)}\right) = \left\| q(k+1;\tau^{(k-l:k)}) - b^{(k+1)} \right\|_2. \tag{7}$$

3.3.1 A Predictive Tracking Algorithm

Using the cost function (7), we can now give a high-level description of our approach for tracking droplets in monocular videos (see Algorithm 1). The loop at line 5 accumulates a set of candidate matches for τ in S and corresponding edges are added to $\mathcal{E}^{(k)}$ in line 8. As in (6), the set $\mathcal{E}^{(k)}$ of allowable edges (i.e., those with cost below the tracking threshold $\varepsilon_{\text{track}}$) dictates the sparsity structure of the assignment problem. The final loop at line 10 extends the paths $\mathcal{T}^{(1:k)}$ covered by $M^{(k)}$ once the matched targets $b^{(k+1)} \in \mathcal{B}^{(k+1)}$ have been determined. Finally, at line 13, the set of paths $\mathcal{T}^{(1:k+1)}$ is extended with new singleton paths $\langle b^{(k+1)} \rangle$ containing the targets in $\mathcal{B}^{(k+1)}$ not covered by $M^{(k)}$. Figure 1 shows the trajectories harvested during a single experiment. We stress

Algorithm 1 Tracking and assembling trajectories in the image plane.

Input: Set of targets in each frame $\{\mathcal{B}^{(k)}\}_{k=1}^{N}$
Output: Set $\mathcal{T}^{(1:N)}$ of (possibly incomplete) paths

$\mathcal{T}^{(1:1)} \leftarrow \left\{ \langle b \rangle \mid b \in \mathcal{B}^{(1)} \right\}$
for $k = 1 : N-1$ **do**
  Initialize $\mathcal{E}^{(k)} \leftarrow \emptyset$
  Initialize $\mathcal{T}^{(1:k+1)} \leftarrow \emptyset$
  for each path $\tau \in \mathcal{T}^{(1:k)}$ **do**
    Predict $\hat{b}_{\tau}^{(k+1)} = q(k+1; \tau^{(k-l:k)})$ from τ, if possible.
    Find $S \subseteq \mathcal{B}^{(k+1)}$ within distance $\varepsilon_{\text{track}}$ of $\hat{b}_{\tau}^{(k+1)}$
    Update $\mathcal{E}^{(k)} \leftarrow \mathcal{E}^{(k)} \cup \{ \langle \tau, b^{(k+1)} \rangle \mid b^{(k+1)} \in S \}$
  $M^{(k)} = \text{SolveSparseAP}(\mathcal{T}^{(1:k)}, \mathcal{B}^{(k+1)}, \text{dist}, \mathcal{E}^{(k)})$ with dist as in (7).
  for each edge $\langle \tau, b^{(k+1)} \rangle \in M^{(k)}$ **do**
    Append $b^{(k+1)}$ to τ and add it to $\mathcal{T}^{(1:k+1)}$
    Mark $b^{(k+1)} \in \mathcal{B}^{(k+1)}$ as *matched*.
  Augment $\mathcal{T}^{(1:k+1)}$ with the *unmatched* blobs remaining in $\mathcal{B}^{(k+1)}$ as singleton paths.
return $\mathcal{T}^{(1:N)}$

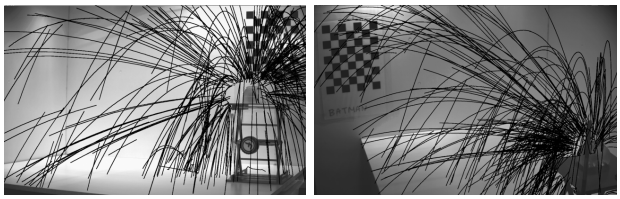

Fig. 1: Trajectories extracted during from a single experiment from each camera.

that this algorithm can only construct droplet trajectories in the image plane and is unable to reconstruct 3D trajectories of these droplets. We return to 3D reconstruction in the next section.

4 Stereo Reconstructions of 3D Trajectories

We now turn our attention to the problem of estimating 3D trajectories given the 2D paths captured by two different cameras. We first consider the well-known problem of using a set of n measurements

$$M = \{ \langle \varsigma_i, t_i, \mathbf{x}_i \rangle \mid i = 1 \ldots n \} \tag{8}$$

captured by calibrated cameras to reconstruct the location of an object. We use the pinhole camera model, with the understanding that images from a non-pinhole camera need to be pre-processed to account for the distortions introduced by the lens. In (8), $\varsigma_i \in \mathbb{Z}$, $t_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathbb{R}^2$ represent the camera id, timestamp and location in the image plane corresponding to the ith measurement, respectively. Each camera ς defines a unique projection operator $\varphi_{\varsigma} : \mathbb{R}^3 \to \mathbb{R}^2$ that maps the subset of $\mathbb{R}^3$ captured by the camera ς to the image plane. The projection operator φ_{ς} can be represented using the 3×4 projection matrix $\mathbf{P}_{\varsigma}$ associated with camera ς:

$$\begin{pmatrix} \varphi_{\varsigma}(\boldsymbol{\lambda}) \\ 1 \end{pmatrix} s = \mathbf{P}_{\varsigma} \begin{pmatrix} \boldsymbol{\lambda} \\ 1 \end{pmatrix}. \tag{9}$$

For brevity, we refer to the projection operator φ_{ς_i} and the camera matrix $\mathbf{P}_{\varsigma_i}$, corresponding to the camera ς_i of measurement $m_i \in M$, as φ_i and $\mathbf{P}_i$, respectively.

The simplest 3D reconstruction problem consists of estimating the unknown location $\boldsymbol{\lambda}$ of an object when all the measurements M are taken simultaneously at time t ($\forall m_i \in M, t_i = t$) by at least two different cameras ($\exists m_i, m_j \in M : \varsigma_i \neq \varsigma_j$)[1]. Generally speaking, $\mathbf{x}_i$ is a noisy observation of $\varphi_i(\boldsymbol{\lambda})$ due to sensing limitations of physical cameras, limitations of the detection routines and calibration errors. This suggests that the *reconstruction problem* consists of finding the location $\boldsymbol{\lambda}$ that best explains the observations $\mathbf{x}_i$. We formalize this notion by introducing a misfit function $\mathrm{e}(\boldsymbol{\lambda}, M)$ that quantifies the discrepancy between the estimated location $\boldsymbol{\lambda}$ and the set of measurements M. One such misfit function is the squared Euclidean distance between the observations and the expected locations of the projections given the operators φ_i, i.e.:

$$\mathrm{e}(\boldsymbol{\lambda}, M) = \sum_i \| \mathbf{x}_i - \varphi_i(\boldsymbol{\lambda}) \|^2. \tag{10}$$

It is straightforward to minimize the above misfit function by solving the following *least squares* problem

$$\hat{\boldsymbol{\lambda}} = \operatorname*{argmin}_{\boldsymbol{\lambda}} \sum_i \| \mathbf{x}_i - \varphi_i(\boldsymbol{\lambda}) \|^2. \tag{11}$$

It is worth remembering that the misfit measure (10) is biased if the aspect ratio of a camera is not $1:1$. This, however, can be easily remedied when internal camera parameters are known. The projection operators φ_i (9) are nonlinear and so the least squares problem described by (11) is also nonlinear. We can derive a linear approximation to this least squares problem by observing that vectors

$$\begin{pmatrix} \varphi_i(\boldsymbol{\lambda}) \\ 1 \end{pmatrix} \text{ and } \mathbf{P}_i \begin{pmatrix} \boldsymbol{\lambda} \\ 1 \end{pmatrix}$$

differ only by a scaling factor, suggesting that their cross product is 0:

$$\left[\begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix} \right]_{\times} \mathbf{P}_i \begin{pmatrix} \boldsymbol{\lambda} \\ 1 \end{pmatrix} = 0 \tag{12a}$$

$$\left[\begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix} \right]_{\times} \mathbf{P}_{i,1:3}\, \boldsymbol{\lambda} = - \left[\begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix} \right]_{\times} \mathbf{P}_{i,4}. \tag{12b}$$

In (12), $[\cdot]_{\times}$ is the skew-symmetric matrix representation of the cross product (as seen in [25]) and the matrices $\mathbf{P}_{i,1:3} \in \mathbb{R}^{3\times 3}$, $\mathbf{P}_{i,4} \in \mathbb{R}^{3\times 1}$ refer to the first three and the fourth column of matrix $\mathbf{P}_i$, respectively. The system of equations (12b) has rank 2, which confirms that a single measurement is insufficient to reconstruct the location $\boldsymbol{\lambda} \in \mathbb{R}^3$. In fact, the third equation of

1. This problem is equivalent to having non-simultaneous measurements of a static object. However, because the objects considered in this paper *are* moving, ours is a more convenient formulation.

the system (12b) is always a linear combination of the first two and is, therefore, redundant. We can rewrite (12b) as

$$\mathbf{Q}_i\boldsymbol{\lambda} = \mathbf{q}_i, \text{ with} \tag{13a}$$

$$\mathbf{Q}_i := \mathbf{I}_{2\times 3}\left[\begin{pmatrix}\mathbf{x}_i\\1\end{pmatrix}\right]_\times \mathbf{P}_{i,1:3} \in \mathbb{R}^{2\times 3} \text{ and} \tag{13b}$$

$$\mathbf{q}_i := -\mathbf{I}_{2\times 3}\left[\begin{pmatrix}\mathbf{x}_i\\1\end{pmatrix}\right]_\times \mathbf{P}_{i,4} \in \mathbb{R}^{2\times 1}. \tag{13c}$$

For every measurement $m_i \in M$ we can derive a new system of equations (13). Combining them, we can build the overdetermined system

$$(\mathbf{Q}_1^T \dots \mathbf{Q}_n^T)^T\boldsymbol{\lambda} = (\mathbf{q}_1 \dots \mathbf{q}_n)^T. \tag{14}$$

The system of equations (14) has three unknowns ($\boldsymbol{\lambda} \in \mathbb{R}^3$) and two equations per measurement. The least squares approximation of this system is a reasonable estimate of the unknown location $\boldsymbol{\lambda}$. The solutions of (11) and (14) do not necessarily agree if the measurements are noisy. Specifically, the misfit associated with (11) is related to the distance between the projections of the estimated 3D location and the observations in the image plane; whereas the misfit in (12a) depends upon both the angle between vectors $\begin{pmatrix}\mathbf{x}_i\\1\end{pmatrix}$ and $\mathbf{P}_i\begin{pmatrix}\boldsymbol{\lambda}\\1\end{pmatrix}$ and their lengths. Nonetheless, both solutions are sufficiently similar for our purposes.

We can use (14) to reconstruct the $3D$ trajectories of the droplets in our original problem provided at least two cameras are used. Let ζ_1 and ζ_2 be the cameras recording the event, and $\mathcal{T}_{\zeta_1}$ and $\mathcal{T}_{\zeta_2}$ the corresponding image plane trajectories as computed by Algorithm 1. For convenience, we augment each trajectory $\tau = \langle b_1, \dots, b_l\rangle \in \mathcal{T}_\varsigma$, $\varsigma \in \{\zeta_1, \zeta_2\}$ with the timestamps t, centroid coordinates $\mathbf{x}$ and the camera label ς corresponding to each blob:

$$\tau^* := \{\langle \varsigma_i = \varsigma, t_i, \mathbf{x}_i\rangle \mid i = 1 \dots |\tau|\}$$
$$M_\varsigma := \{\tau^* \mid \tau \in \mathcal{T}_\varsigma\}.$$

If we know that the measurement sets $\tau_1^* \in M_{\zeta_1}$ and $\tau_2^* \in M_{\zeta_2}$ correspond to the same droplet, we can use (14) with the combined measurement set $\tau_{1\cup 2}^* = \tau_1^* \bigcup \tau_2^*$ to retrieve the $3D$ locations at every common timestamp. Although the pairings $\tau_1^* \in M_{\zeta_1}$ and $\tau_2^* \in M_{\zeta_2}$ are not known beforehand (see Figure 1), equation (14) can be used to find them. We sketch this procedure in Algorithm 2.

The output C of algorithm 2 is the desired set of trajectories consisting of three-dimensional world coordinates $\hat{\boldsymbol{\lambda}}_t$, the corresponding timestamps t and the errors e_t. If enough world coordinates $\hat{\boldsymbol{\lambda}}_t$ are known for a given trajectory $c \in C$, we can use them to study the function describing that trajectory. Given the estimated world coordinates $\hat{\boldsymbol{\lambda}}_t$ and a function family $\mathcal{L}_{\boldsymbol{\alpha}}$ which depends on a set of parameters $\boldsymbol{\alpha}$, we can compute the optimal set of parameters $\boldsymbol{\alpha}$ for these measurements by minimizing the least squares error:

$$\mathrm{e}(\boldsymbol{\alpha}, c) = \sum_{\langle t, \hat{\boldsymbol{\lambda}}_t, e_t\rangle \in c} \|\mathcal{L}_{\boldsymbol{\alpha}}(t) - \hat{\boldsymbol{\lambda}}_t\|^2. \tag{15}$$

This approach is followed in [34].

Algorithm 2 Reconstruction of 3D trajectories from simultaneous measurements.

Input: Collections of augmented measurements M_{ζ_1} and M_{ζ_2} captured by cameras ζ_1 and ζ_2.
Output: Set C of trajectories in three dimensions, set $M_{1\cup 2}$ of combined measurements.

$\mathcal{E} \leftarrow \emptyset$
for $\tau_1^* \in M_{\zeta_1}$ **do**
  for $\tau_2^* \in M_{\zeta_2}$ **do**
    $T \leftarrow$ set of common timestamps in τ_1^* and τ_2^*.
    if $|T| > k$ **then**
      $c_{\tau_1^*,\tau_2^*} \leftarrow \emptyset$
      for $t \in T$ **do**
        Select $m_1 \in \tau_1^*$, $m_2 \in \tau_2^*$ for timestamp t.
        Estimate most likely location $\boldsymbol{\lambda}_t$ using (14).
        Compute the reprojection error e_t using (10).
        if $e_t > e_{max}$ **then**
          Classify the measurement as an outlier
        else
          Append tuple $\langle t, \boldsymbol{\lambda}_t, e_t\rangle$ onto $c_{\tau_1^*,\tau_2^*}$
    if $\left|c_{\tau_1^*,\tau_2^*}\right| > k$ **then**
      Define $\mathrm{dist}(\tau_1^*, \tau_2^*) :=$ average of e_t.
      Update $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{\langle \tau_1^*, \tau_2^*\rangle\}$.
$M = \mathrm{SolveSparseAP}(M_{\zeta_1}, M_{\zeta_2}, \mathrm{dist}, \mathcal{E})$ with dist as above.
$M_{1\cup 2} \leftarrow \bigcup_{\langle \tau_1^*, \tau_2^*\rangle \in M} \{\tau_1^* \cup \tau_2^*\}$
$C \leftarrow \{c_{\tau_1^*,\tau_2^*} \mid \langle \tau_1^*, \tau_2^*\rangle \in M\}$

5 Improved 3D Reconstruction using Motion Priors

The work described in the previous section is similar to the *tracking-reconstruction* ideas discussed in [35]. It assumes that 1) the cameras recording the droplets are synchronized and 2) each droplet is visible in at least two cameras before its 3D location can be estimated. In the following sections, we develop a method that relaxes these assumptions.

5.1 3D Reconstruction Using Linear Motion Priors

Park et. al. [30] have shown that if the motion of an object can be represented as an unknown linear combination of some set of known basis functions and its motion is not correlated with the motion of the camera, it is possible to reconstruct the motion of the object. We are able to exploit this observation since the motion of droplets is not correlated with our cameras, which are stationary. Let $\Theta = (\theta_1, \dots, \theta_k)$ be the known basis functions. If we hypothesize that the target can be approximated by a linear combination of the functions in Θ then

$$\boldsymbol{\lambda}_i \approx \sum_{j=1}^{k} c_j\theta_j(t_i) = \Theta_i\mathbf{C} \tag{16}$$

for some coefficients $\mathbf{C} = (c_1, \dots, c_k)^T \in \mathbb{R}^k$. Using the linearization (13), we can then find the parameters $\mathbf{C}$ by solving the following linear least squares problem:

$$\mathbf{Q}\bar{\Theta}\mathbf{C} = \mathbf{q}, \tag{17}$$

where

$$\mathbf{Q} := \begin{pmatrix} \mathbf{Q}_1 & 0 & \dots & 0 \\ 0 & \mathbf{Q}_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \mathbf{Q}_n \end{pmatrix},$$

$$\bar{\Theta} := \begin{pmatrix} \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_n \end{pmatrix} \quad \text{and} \quad \mathbf{q} := \begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_n \end{pmatrix}.$$

The effects of drag over very short periods of time are small, in which case, the trajectory can be approximated by a quadratic polynomial over small intervals. This observation allows us to use Park's formulation to estimate the $3D$ locations corresponding to single measurements. The algorithm 2 can be updated as follows: if two or more measurements are known at time t_i, reconstruct the 3D location as before, otherwise, use equation (17) to estimate the polynomial coefficients $\mathbf{C}$ using the measurements between $t_i - \Delta t$ and $t_i + \Delta t$, accept the match if the reprojections are sufficiently close to the known measurements, and estimate $\boldsymbol{\lambda}_i$ by evaluating the polynomial at time t_i. The matrices Θ_i corresponding to a quadratic trajectory are given by:

$$\Theta_i = \begin{pmatrix} t_i^2 & 0 & 0 & t_i & 0 & 0 & 1 & 0 & 0 \\ 0 & t_i^2 & 0 & 0 & t_i & 0 & 0 & 1 & 0 \\ 0 & 0 & t_i^2 & 0 & 0 & t_i & 0 & 0 & 1 \end{pmatrix}.$$

5.2 Nonlinear Reconstruction of 3D Motion from Unsynchronized Cameras

We could use equation 15 and the individual 3D reconstructions (either the output of algorithm 2 or the extension discussed in the previous section) to study the flight trajectories. Doing this, however, incurs two sources of error, namely:

- The stereo reconstruction error increases not only with the errors in the 2D measurements, but also with the distance between the target and the cameras. This uneven propagation of the error is difficult to model when studying the flight trajectories, but by ignoring it, we risk assigning too much importance to a reconstructed point, which may be highly contaminated by error when solving (15).
- The reconstruction in section 5.1 assumed a polynomial motion for the points where a direct 3D reconstruction was not feasible and can lead to unrealistic reconstructions if the Δt parameter is not carefully chosen.

These sources of error may or may not be significant enough to affect the estimation of the parameters $\boldsymbol{\alpha}$ in (15). However, it is straightforward to eliminate them, by combining the trajectory reconstruction and the parameter estimation steps. We rewrite (15) as follows:

$$\mathrm{e}(\boldsymbol{\alpha}, c) = \sum_{\langle \varsigma_i, t_i, \mathbf{x}_i \rangle \in c} \| \varphi_i(\mathcal{L}_{\boldsymbol{\alpha}}(t_i)) - \mathbf{x}_i \|^2 \tag{18}$$

where $\mathbf{x}_i$ represents the coordinates in the image plane for the detected target, t_i is the timestamp of the measurement, φ_i is the projection operator corresponding to camera $\varsigma_i \in \{\zeta_1, \zeta_2\}$ and the set of measurements c consists of all the measurements $\langle \varsigma_i, t_i, \mathbf{x}_i \rangle$ recorded for a given target.

With this approach, it is no longer necessary to estimate the 3D locations *a priori* in order to start the optimization process to minimize (10). Instead, the parameter vector $\boldsymbol{\alpha}$ is estimated directly from the $2D$ measurements using some nonlinear solver, bypassing any error introduced by the 3D reconstruction. Once the optimal $\boldsymbol{\alpha}$ is found, it is straightforward to compute the optimal 3D locations $\boldsymbol{\lambda}_i \approx \mathcal{L}_{\boldsymbol{\alpha}}(t_i)$. We cannot bypass Algorithm 2 entirely, because we still need to match the trajectories from camera ζ_1 to the trajectories from ζ_2.

5.3 A Note About Outliers

When dealing with a real detector, such as the one used in [34], spurious measurements may be erroneously added to a trajectory. These outliers do not satisfy the motion model of the droplets, therefore, we need to detect and remove them to obtain accurate results. In particular, the parabolic tracker from Section 3.3 tends to produce outliers at the beginning of the trajectories, when there is not enough data to find the quadratic coefficients. Outliers may also appear near the end of the trajectory, when the tracker, unable to find a new measurement, may instead accept some blob near the predicted location. Most notably, this also occurs when the liquid droplet impacts a surface, as the tracker may try to follow the splashing particles before giving up (Figure 2, top left).

Both the linear (17) and the nonlinear (18) methods are sensitive to outliers. In the linear case, they can be handled with Random Consensus Sampling (RANSAC), as we only need to solve a small system of equations for each guess. For the nonlinear case (18), RANSAC is prohibitively expensive, as it would require to solve a nonlinear optimization problem for each candidate subset. Instead, we follow a 2-pass approach to handle outliers. Using the Huber loss function [36]:

$$H_\delta(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| < \delta \\ \delta(|x| - \frac{\delta}{2}) & \text{otherwise} \end{cases} \tag{19}$$

We rewrite the objective function (18) as

$$\mathrm{e}_\delta(\boldsymbol{\alpha}, c) = \sum_{\langle \mathbf{x}_i, \mathbf{P}_i, t_i \rangle \in c} H_\delta(\mathbf{x}_i - \varphi_i(\mathcal{L}_{\boldsymbol{\alpha}}(t_i))). \tag{20}$$

In the *robustified* objective function (20), outliers are given less weight than in (18) and therefore have less influence in the minimizer $\hat{\boldsymbol{\alpha}}$. Figure 2 (top) illustrates this: the "splashing" section of the trajectory (red) has limited influence on the estimated function $\mathcal{L}_{\hat{\boldsymbol{\alpha}}}$. We then discard the measurements $\mathbf{x}_i$ at distance greater than δ from $\varphi_i(\mathcal{L}_{\hat{\boldsymbol{\alpha}}}(t_i))$ and repeat the optimization, using the parameters $\hat{\boldsymbol{\alpha}}$ previously computed as initial guess. Figure 2 (bottom) shows the final function $\mathcal{L}_{\hat{\boldsymbol{\alpha}}}$ (black line) and the measurments used in the second pass (green line).

6 Blood Droplet Flight Models

We compare four parametrized flight models in our experiments. The parameters in each model correspond to distinct assumptions about the dynamics of the flight.

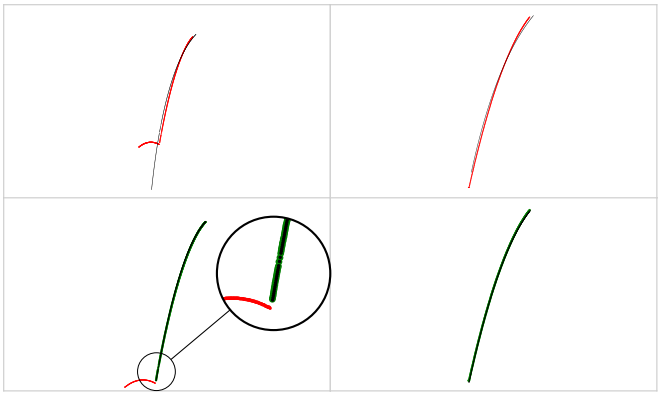

Fig. 2: Outlier detection: a single trajectory (red) is recorded by the cameras ζ_1 (left) and ζ_2 (right). A first pass fits the ODE parameters (black) using a robustified loss function (top). The outliers are then discarded and a second pass, using only the inliers (green), computes the final fit (bottom).

1) *Quadratic drag.* Using Newton's second law of motion and a quadratic model of drag, we can derive the differential equation

$$\begin{aligned} \dot{\mathbf{v}} &= -K\|\mathbf{v}\|\mathbf{v}+\mathbf{g}, \\ \boldsymbol{\alpha} &= (\boldsymbol{\lambda}_0, \mathbf{v}_0, K) \in \mathbb{R}^7. \end{aligned} \tag{21}$$

2) *Linear drag*. For objects moving at low speed in a non-turbulent fluid, a drag force linearly proportional to speed—but with opposite direction—may be more appropriate [37]. This yields the differential equation

$$\begin{aligned} \dot{\mathbf{v}} &= -K\mathbf{v}+\mathbf{g}, \\ \boldsymbol{\alpha} &= (\boldsymbol{\lambda}_0, \mathbf{v}_0, K) \in \mathbb{R}^7 \end{aligned} \tag{22}$$

Unlike (21), this equation admits a closed solution for the position $\boldsymbol{\lambda}$, namely

$$\begin{aligned} \mathbf{v}_\infty &= \frac{\mathbf{g}}{K}, \\ \boldsymbol{\lambda}(t) &= \mathbf{v}_\infty t + \frac{1}{K}(\mathbf{v}_0 - \mathbf{v}_\infty)(1-e^{-Kt}) + \boldsymbol{\lambda}_0, \\ \mathbf{v} &= \dot{\boldsymbol{\lambda}}. \end{aligned}$$

Notice that the drag coefficients K in (21) and (22) are not necessarily identical for distinct droplets. While we should be able to find a suitable drag coefficient K for each droplet using the algorithms described in this paper if the trajectories are recorded, such estimation would be unfeasible in the inverse problem, when only the final measurements, but not the trajectories, are available[2].

3) *No drag*. When the effects of aerodynamic drag are completely ignored, the motion of a droplet influenced only by gravity is described by

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\alpha}}(t) &= \boldsymbol{\lambda}_0 + \mathbf{v}_0 t + \tfrac{1}{2}\mathbf{g}t^2, \\ \boldsymbol{\alpha} &= (\boldsymbol{\lambda}_0, \mathbf{v}_0) \in \mathbb{R}^6. \end{aligned} \tag{23}$$

4) *Quadratic polynomial*. A trivial generalization of the *no drag* model is to approximate the flight with a generic quadratic polynomial. This model corresponds to the unrealistic scenario of the drag force being unknown but constant, in both direction and magnitude:

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\alpha}}(t) &= \boldsymbol{\lambda}_0 + \mathbf{v}_0 t + \mathbf{a}t^2, \\ \boldsymbol{\alpha} &= (\boldsymbol{\lambda}_0, \mathbf{v}_0, \mathbf{a}) \in \mathbb{R}^9 \end{aligned} \tag{24}$$

The parameters in the model (24) can be estimated using Park's method as described in 5.1. The polynomial model can also be used as an initial estimate when solving non-linear equations iteratively.

2. After all, in the forensics application, if the trajectories were known, there would be no need to estimate the point of origin.

7 Results

We have conducted a series of real and synthetic experiments to evaluate the performance of our tracker. We now summarize our results. The details of the real experiments and the physical set up needed to carry out the real experiments can be found in [34] and in Sec. 7.2.1. We do not have the ground truth trajectories for real experiments. This is due to our inability to capture high-speed, high-precision 3D measurements of the moving droplets. In fact, the main motivation behind this work was to develop a system capable of capturing these 3D measurements, thereby providing a means to study high speed 3D fenomena. For this reason, it is hard to estimate the errors in the reconstruction. To overcome this limitation, we designed synthetic experiments with known ground truth, under the assumption that our reconstruction strategy will perform similarly in both synthetic and real scenarios.

7.1 Synthetic Experiments

We simulate the motion of a spherical object moving under the effects of drag and gravity. Equation (21) is used to model the 3D locations of the center of mass. These locations are then projected to the image planes of two virtual cameras, simulating a data capture at 1300 frames per second. The imaging parameters of our virtual stereo camera pair were chosen to match those of our physical cameras. This system allows us to to "capture" the 2D projections of the moving object in cameras ζ_1 and ζ_2 at each instant, providing us with true 2D trajectories of the moving object as seen in each image.

A synthetic experiment, however, is not affected by measurement errors the same way than the real experiments. Equations (21), (22), (23) and (24) model the movement of a single point within the droplet: its center of mass. For each droplet, however, a real camera is only able to capture a blob, with no indication of which of the pixels within a blob may correspond to the projection of the center of mass. Lacking this information, we nonetheless need to estimate a location to evaluate the misfit function (18). Knowing that it is unlikely for the center of mass to be close to the edge of the droplet, we chose to approximate its projection with the centroid of the blob. Together with the possibility of miss-associations during 2D tracking—which may cause us to select the wrong blob altogether—, this approximation becomes a source of measurement errors not present in the synthetic experiments.

Another source of errors arise from the camera calibration itself. A real camera is most likely not perfectly calibrated.

This means that the projection function φ, that translates points in 3D space to the exact location in the image plane, is but an approximation of an idealized, but unknown function. Combined with the finite resolution of a camera, there is no reason to expect the theoretical projection $\varphi(\boldsymbol{\lambda})$ of a point $\boldsymbol{\lambda} \in \mathbb{R}^3$ to coincide with the actual measurement $\mathbf{x}$ recorded by a real camera.

Both sources of error are present in the real world experiments, but not in the synthetic ones. In order to evaluate the quality of the reconstruction techniques, we need to simulate these measurement errors. We do so by adding a Gaussian perturbation to the projections: $\mathbf{x}_i = \varphi_i(\mathcal{L}_{\boldsymbol{\alpha}}(t_i)) + e_i,\ e_i \sim N(0, \sigma \mathbf{I}_{2\times 2})$. The addition of a Gaussian perturbation with any particular σ (and the implicit assumption that the measurement errors are independent and identically distributed) is somewhat arbitrary, but serves to compare the sensitivity to measurement errors of the different 3D reconstruction techniques. For the purposes of this comparison, we chose a value of $\sigma = 1.83$ (pixels). We discuss our selection of this value in Sec. 7.3.

Finally, the simulated cameras described above are synchronized, producing one measurement per camera per time step. We know that this is not necessarily true in real experiments: targets may remain undetected (or miss-associated) in one camera but not the other. To model this, we introduce a *detection ratio* to the simulation: the probability that a single measurement will be "recorded" by a virtual camera. We show that even with unrealistically low detection ratios we are able to reconstruct the trajectory with reasonable accuracy.

7.1.1 Accuracy of the parameter estimation

In the first experiment, we used the synthetic data to estimate the parameter vector $\boldsymbol{\alpha}$ corresponding to each trajectory, and to compare the estimate with the known ground truth. This can provide insight into how sensitive is the reconstruction to the image noise and to the detection ratio.

Table 2 shows the result with 100 trajectories, each evolved for 1 second at a simulated frame rate of 1300 frames per second, for different detection ratios. For each row, we list the detection ratio, the average number of measurements per path, the mean error in the estimation of the initial location $\boldsymbol{\lambda}_0$, velocity $\mathbf{v}_0$ and drag coefficient K, and the mean distance between the reconstruction and the ground truth trajectory. In Table 3, normal noise with $\sigma = 1.83$ was added to the measurements. Figure 3 shows the reconstruction for some of the noisy trajectories from 5% of the measurements. Even with so few points, the reconstructed trajectory is remarkably similar to the ground truth functions.

It is clear from table 2 that, even with a very high rate of misdetections, the unsynchronized reconstruction algorithm can retrieve both the flight parameters and the 3D locations with remarkable accuracy. However, table 3 shows that if the image plane measurements are very noisy, a higher number of measurements is required to obtain a low reconstruction error. Even with 50% detection ratio, the average error was at about 6.8 mm, about ten times higher than in the noiseless scenario, and increases rapidly if less measurements are available. This result stresses the need of maximizing the number of usable measurements.

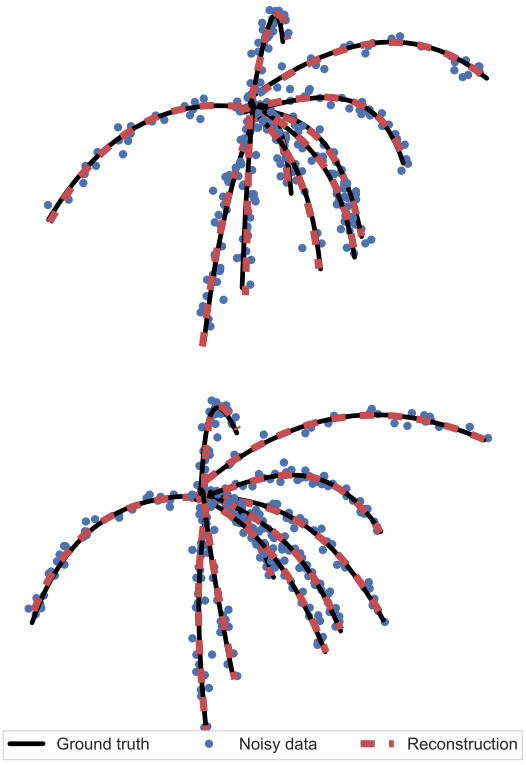

Fig. 3: Re-projection into the two virtual cameras of some simulated trajectories. The dotted line represents the noisy data, the solid line is the ground truth, and the dashed line shows the trajectory reconstructed from the noisy data.

TABLE 4: Average deviation from the ground truth, in cm, using different reconstructions from noisy data, with detection ratio of 0.1, 0.2, 0.3, 0.5, 0.7 and 1.0. 100 random trajectories of 1s duration with normal noise ($\sigma = 1.83$) were used.

Detection ratio	Triangulation	3D optimization	Unsynchronized
0.1	11.9	5.258	2.012
0.2	12.14	2.501	1.202
0.3	12.13	1.676	0.8803
0.5	12.15	0.9874	0.682
0.7	12.17	0.7351	0.556
1.0	12.14	0.471	0.4276

7.1.2 Comparison of synchronized and unsynchronized reconstruction

Given this result, it is interesting to evaluate how the unsynchronized reconstruction algorithm presented in 5.2 compares with the traditional 3D reconstruction methods described in section 4.

Table 4 compares the 3D reconstruction of 100 synthetic trajectories of 1s in duration. The synchronized reconstruction solves the location of each point independently, using the triangulation equation (14) from section 4. Observe the very high average deviation from the ground truth. As expected, the error does not improve with the detection ratio. The 3D optimization fits the model (21) to the synchronized 3D reconstruction and was the approach followed by [34]. It produces a significant improvement over equation (14) in most

TABLE 2: Accuracy of the estimated parameters from noiseless data, compared with the ground truth. The errors are given in terms of the average distance between the estimated parameters and the true parameters, the average error displays the mean distance between the reconstructed trajectory and the ground truth.

Detection ratio	**Avg. number of measurements per path**	λ_0 **error (cm)**	v_0 **error, (m/s)**	K **error**	**Avg. error (cm)**
0.5	1303.69	1.17e-01	9.93e-03	8.29e-11	6.17e-02
0.1	261.09	1.75e+00	1.18e-01	1.93e-09	1.08e+00
0.05	130.30	3.99e+00	2.89e-01	2.77e-09	2.46e+00

TABLE 3: Like table 2, but with normal noise of $\sigma = 1.83$ added to the measurements.

Detection ratio	**Avg. number of measurements per path**	λ_0 **error (cm)**	v_0 **error, (m/s)**	K **error**	**Avg. error (cm)**
0.5	1303.69	1.24e+00	7.14e-02	2.03e-03	6.82e-01
0.1	261.09	3.63e+00	2.14e-01	4.09e-03	2.01e+00
0.05	130.30	6.06e+00	4.03e-01	5.51e-03	3.59e+00

cases, though it still depends heavily on the detection ratio.

Finally, the unsynchronized reconstruction proposed in section 5.2, using the *quadratic drag model* (21) produces a significantly better approximation to the ground truth. Even in the absence of full stereo information, such as in the first row, our unsynchronized reconstruction method was able to use the single measurements to estimate the parameters, producing significantly better approximations than the 3D optimization with incomplete data.

The last row is particularly interesting. Even if all the measurements are synchronized, the two optimization methods produced much better results than the traditional triangulation approach, which suggests that our unsynchronized method should be preferred over the traditional methods whenever prior knowledge of the motion is available.

7.1.3 Predictive power

In 3.3, a $2D$ quadratic polynomial approximation was employed to track droplets in the image plane. It was found that the quadratic model was able to accurately predict the expected locations of the droplets within a very small time window. It was also noticed that the quadratic predictions broke down quickly after a few frames without a measurement, or if too many measurements were used to estimate the model.

An accurate flight model should be able to predict future locations of the droplet from only a small set of noisy measurements. Using synthetic data, table 5 compares the accuracy of the reconstruction if only a small window of time is available. The accuracy is measured as the average deviation from the ground truth for the entire simulation time. This allows us to evaluate the predictive capacity of each simplified flight model, that is, how precisely each model allows us to extrapolate from a small set of measurements. We can see that the ODE models behaved significantly better than the *quadratic polynomial* model. The *no drag* model also behaved significantly better than the *quadratic polynomial* model in the noisy case for the first two columns. Figure 4 suggests why this happens: having more degrees of freedom, the *quadratic polynomial* model may overfit the noise in the data, resulting in a very inaccurate extrapolation. This is consistent with the prediction breakdown observed in [32]. Without noise (Figure 5), both ODEs were able to accurately reconstruct the entire trajectory using only the first $0.1s$ of flight. The polynomial models also behaved much better with noiseless data, but they were still significantly worse than the ODEs.

It is not surprising that the *quadratic drag model* is the

TABLE 5: Prediction accuracy, with noise (top) and without noise (bottom). The values represent the mean distance, in centimeters, between the ground truth and the estimated model over the entire trajectory, when only measurements corresponding to the first $0.1s, 0.2s, 0.3s$ and $0.5s$ of flight are used.

With noise				
Model \ time (s)	0.1	0.2	0.3	0.5
no drag	1.3e+02	1.1e+02	1e+02	1.1e+02
constant force	5.8e+02	1.6e+02	93	41
quadratic drag	20	6.4	2.8	1.3
linear drag	35	16	10	4.1
Without noise				
Model \ time (s)	0.1	0.2	0.3	0.5
no drag	1.3e+02	1.1e+02	1e+02	1.1e+02
constant force	2.1e+02	1.2e+02	85	40
quadratic drag	0.062	0.062	0.062	0.062
linear drag	22	14	9.1	3.6

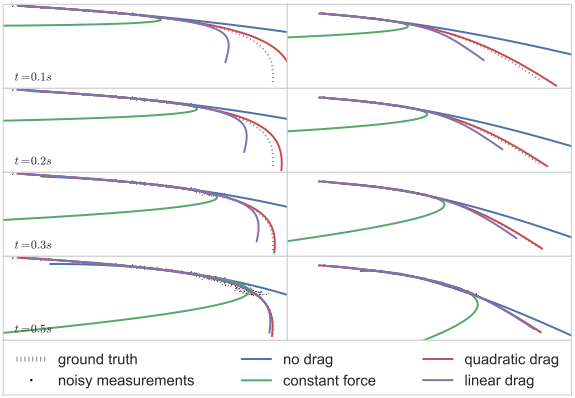

Fig. 4: Reconstruction of a trajectory from a small amount of noisy data ($\sigma = 1.83$). Note how, in the first row, the *quadratic polynomial* model overfits the available data, causing it to behave extremely badly outside of the interval.

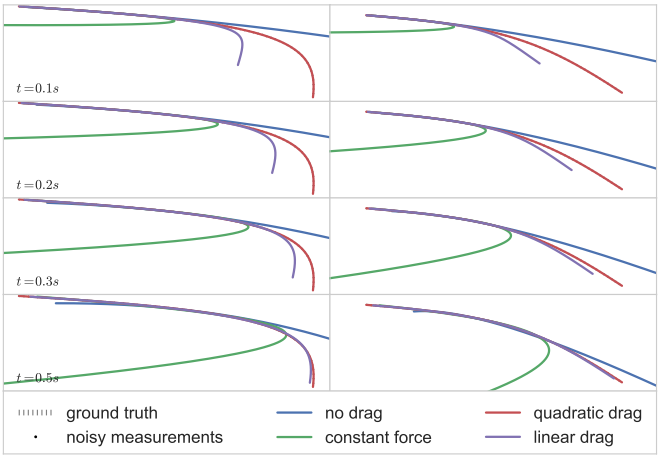

Fig. 5: Like Figure 4, but without any noise added to the data.

TABLE 6: Prediction accuracy for real paths. Because ground truth is not available, the errors are given in terms of pixels in the image plane.

Model \ time (s)	0.1	0.15	0.2
no drag	2	1.4	0.089
constant force	5.7	0.28	0.11
quadratic drag	0.13	0.06	0.049
linear drag	0.13	0.06	0.05

best to explain the synthetic data, after all, the synthetic data was produced by evaluating the quadratic drag equation. These results merely establish that we can compare the accuracy of the flight models by comparing the extrapolations from a subset of the flight with the entire set of 3D points.

This gives us a tool to validate the four flight models with the real data. In real experiments, the true 3D parameters and locations are unavailable, therefore, we can only use the 2D measurements to compare the accuracy of the models.

7.2 Real Experiments

Having established that our unsynchronized reconstruction method produces more accurate results than the triangulation and 3D optimization methods, we attempt to validate our four flight models with data measured by the real experiments.

Figure 6 shows selected frames captured by the stereo camera system during a physical bloodletting experiment. One of the trajectories reconstructed from this experiment is shown in figure 7. The droplet was recorded for about $0.5s$, one of the longest trajectories in that experiment. We fit the model using only the data corresponding to a very small window of time $(0,t)$ and compare the evaluations of the fitted model over the entire interval $(0,0.5)$ with the available measurements. The figure shows that even with very few data points, the ODE models were able to accurately predict the entire trajectory.

A more complete test is summarized in Table 6. This test was run over all the trajectories with at least 100 measurements per camera captured during a single experiment. The errors shown are the average of the distances, in pixels, between the predictions and the measurements in the image plane. With incomplete data, the ODE models behaved significantly better

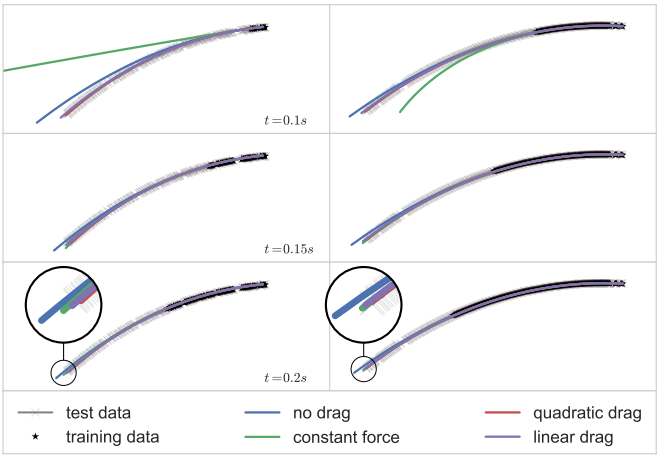

Fig. 7: Reconstruction of a real trajectory. Note that, considering only the first $0.1s$ of flight (top), the ODE models were a near perfect fit for the overall trajectory, while the *quadratic polynomial* model behaved badly. When $0.2s$ are used, the curves are very similar, but the polynomial models still deviate from the data.

Fig. 8: Experimental setup showing ballistic gel containing transfer blood. Experiments are captured using a stereo camera pair capable of recording high-speed video.

than the polynomials, with the quadratic model being slightly better at predicting the future locations of the droplets.

7.2.1 *Lab Setup*

Figure 8 shows our experimental setup. The experiments are set up on a steel table with dimensions roughly $0.9\text{m} \times 3.0\text{m}$. Plywood boards with a white vinyl finish are placed along the two edges opposite to the stereo camera pair. These boards act as walls to contain the splatter and to provide a uniform background for the experiments. The target is a thin latex packet containing 20 ml of transfer blood encased in gelatin designed to approximate human flesh. The target is raised off the table with a lab jack. The paintball gun sits directly behind the target.

Two high-speed cameras, ζ_1 and ζ_2 , protected by sheets of Plexiglas, record the experimental setup from two viewpoints, capturing videos necessary to reconstruct the droplets' trajectories in 3D. Fourteen 500 W work lights illuminate the scene, allowing cameras to capture videos at 1300 frames per second. Because of the high framerate, the cameras capture the 60 Hz oscillations in the alternate current. Combined with the low exposure time, this causes the scene to *appear* poorly illuminated and with changing background, difficulting

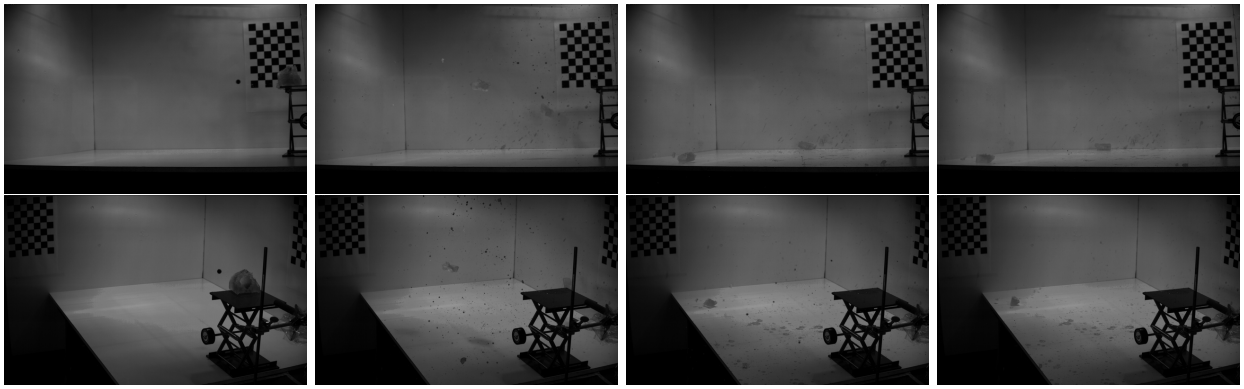

Fig. 6: Selected frames captured by the high-speed stereo camera pair during a porcine blood experiment. The top row shows frames 12, 400, 800 and 1000 captured by camera 1 and the bottom row shows the corresponding frames for camera 2. Frame resolution is 1280×800. The riot ball is visible in the first frame. The darker blobs are blood droplets whereas the translucent blob are images of the ballistic gel. The exposure time for each frame is less than 0.7 milliseconds. For this reason, the frames are grayscale and appear to be poorly illuminated, even though the scene is illuminated by fourteen 500 watts lamps.

the foreground extraction and resulting in a large number of misdetections. A triggering mechanism controls the two cameras to capture synchronized videos. Although the videos themselves are synchronized, the number of misdetections, and the fact that a droplet may be seen in one camera much earlier than it starts registering in the other, causes the number of simultaneous detections to be much smaller than the overall number of detections.

Each experiment begins by recording the calibration rig—a flat checkerboard pattern—moving through the cameras' fields of view. These calibration videos are used to compute each individual cameras' intrinsic and extrinsic parameters [25]. The stereo camera system is calibrated using a check board calibration target using the approach described in [38]. Once calibrated, the cameras are configured to record the experimental area at high speed and the gun is fired. Upon impact with the target, the latex container breaks, causing the gel and the liquid to escape and leaving "bloodstains" on the wall. The process typically takes anywhere from 0.6 to 1.3 seconds, producing up to 1690 grey-scale frames from each camera. Video resolution is 1280×800 pixels.

Our system requires the cameras to remain calibrated for the duration of the experiment. This may not be the case, e.g., if kinetic energy is transferred to the walls or the camera base by recoil from the gun or if the bullet hits a camera after bouncing off the walls. To verify that the calibration is still valid after the recording, we devised the following scheme. We point a laser at the scene and record a pair of videos showing the laser dot moving through the scene. The position of the laser dot in each video frame can then be used to verify the camera pair calibration (see Figure 9). We assume that the location and direction of the cameras do not change during the course of the experiment, i.e, that the extrinsic parameters computed during the calibration process do not change. The laser verification procedure described above is repeated after each experiment to validate the preceding assumption. A failure in the second laser verification signals that at least one of the cameras moved during the experiment. In this case, the experiment is discarded as we cannot use visual cues from the experiment table to compensate for this movement.

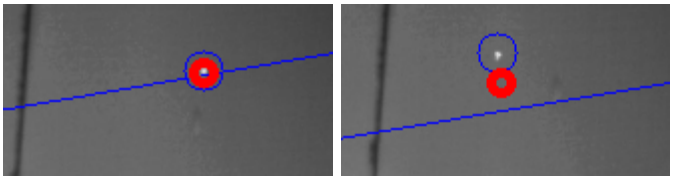

Fig. 9: Calibration verification by shining a laser at the observed region. In these figures, thin blue circles depict the laser dot, blue lines represent the epipolar line corresponding to the the laser dot in the other camera, and the thick red circles show the re-projection of the 3D reconstruction of the laser dot. The left image shows a successful calibration. Here, the epipolar line crosses the laser dot and the re-projection coincides with the original location. The right image shows an inaccurate calibration. One of the cameras has moved, perhaps due to the vibrations generated by the recoil of the gun.

A few pictures of the bloodstains are captured before cleaning up the lab and preparing for the next experiment. These pictures are taken using similar equipment and techniques available to a forensic investigator capturing photographic evidence of bloodstains at a crime scene. The results shown in this section were obtained from 244 trajectories harvested from one bloodletting event.

7.3 A note about σ

The synthetic data is generated by projecting simulated droplet trajectories onto camera views. Gaussian noise is added to these projections to simulate measurement errors in the image space. Specifically, the simulated measurements $\mathbf{x}_i$ in the image space are

$$\mathbf{x}_i = \varphi_i(\mathcal{L}_{\boldsymbol{\alpha}}(t_i)) + e_i,\ e_i \sim N(0, \sigma\, \mathbf{I}_{2\times 2}). \tag{25}$$

Our choice of e_i reflects that in real world experiments, we expect the measurement errors in the image space to follow a bell-shaped distribution. Such assumptions are commonplace in computer vision literature, e.g., see [6] and [5].

Due to our inability to acquire ground truth (3D coordinates of centers of mass of individual droplets), we are unable

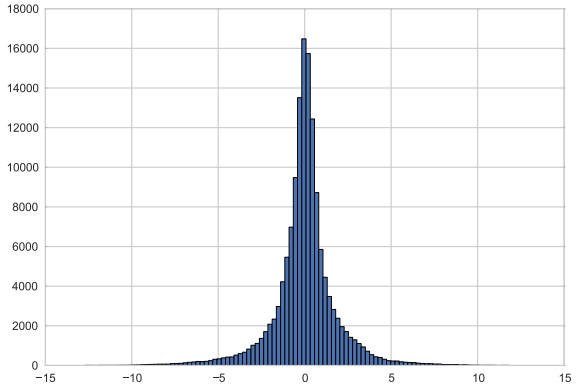

Fig. 10: Distribution of the misfits (in pixels) between measurements and quadratic drag model in the real world experiments. For brevity, this histogram combines the horizontal and vertical components of the misfit vectors.

to characterize the measurement errors associated with our system. Instead, we study the misfits between the real-world measurements and the projections of the fitted quadratic drag model (21). Figure 10 shows a histogram of the misfits between measurements and the quadratic drag model for the real world experiments. This histogram combines the horizontal and vertical components of the misfit vectors. In our experiments, the standard deviation of these misfits was approximately 1.83. Although the misfit histogram shown in Figure 10 is not Gaussian, we observed that bivariate Gaussian noise with $\sigma = 1.83$ introduced error with an average magnitude of 2.30 pixels. This is higher than the average misfit of 1.85 pixels observed in the real data. Therefore, we opted to use $\sigma = 1.83$ for the synthetic experiments, thereby simulating errors comparable, but higher on average, than those observed in real data.

We can also use the experimental data to estimate the detection rates. Again, it is not possible to know for certain how many measurements are missing (due to occlusions, errors in background removal, cameras' field of views limitations, etc) from each trajectory. Instead we can compute the expected number of measurements of a trajectory given its first and last measurements. If the first measurement for trajectory occurs in frame i and the last measurement for this trajectory occurs in frame j then the expected number of measurements for this trajectory is $2(i-j+1)$, i.e., one measurement per frame per camera. We compare this number to the actual number of measurements for each trajectory. For our experiments the detection ratios were between 50% and 99.8%. The overall detection ratio—defined as the ratio between the total number of measurements detected and the expected number of measurements in all trajectories—was 67%.

8 Conclusions

We propose a method capable of estimating the 3D path of a particle from a set of unsynchronized multiview image measurements. Our method is able to reliably reconstruct the 3D trajectory of a particle—given a parametrized model $\mathcal{L}_{\alpha}(t)$ describing its motion—even in the presence of occlusions, misdetections and misclassifications. Unlike other attempts that rely upon motion models to improve 3D estimation accuracy, the proposed method makes no linearity assumptions about the motion model.

We applied the proposed method to the problem of estimating the 3D trajectories of droplets as they move through the air under the influence of drag and gravity. Physical experiments that were carried out to record these droplets are described elsewhere [32]. We also generated a synthetic dataset that faithfully mimics those captured from the physical experimental setup mentioned above. The synthetic dataset allowed us to evaluate the proposed approach under controlled settings. We compared the four motion models described in the previous section and we found that the polynomial motion models fair poorly at the task of extrapolation, i.e., estimating the full 3D trajectory of a particle given a small set of initial measurements. ODE based motion models, on the other hand, correctly extrapolated the 3D trajectory of a particle even in the presence of noisy measurements. We achieved similar results for the real dataset.

Our results summarized in Table 4 are both exciting and encouraging. We conclude that 3D reconstruction via triangulation (using synchronized measurements) often exaggerates measurement errors, affecting 3D trajectory estimation. The proposed technique avoids this issue by solving for both the 3D trajectory of a particle and its motion model concomitantly. We show that the proposed method outperforms the traditional triangulation based 3D trajectory reconstruction approach.

9 Acknowledgements

The research reported herein was supported in part by a grant from the Canadian Police Research Centre. We acknowledge the financial support provided by the National Science and Engineering Council (NSERC) of Canada. We wish to thank R. Murray and P. Prior for setting up the physical experimental setup and carrying out the experiments. We thank F. Gaspari for motivation, encouragement and support.

References

[1] U. Buck, B. Kneubuehl, S. Näther, N. Albertini, L. Schmidt, and M. Thali, "3D bloodstain pattern analysis: Ballistic reconstruction of the trajectories of blood drops and determination of the centres of origin of the bloodstains," *Forensic Sci. Int.*, vol. 206, no. 1-3, pp. 22–28, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0379073810002926

[2] B. T. Cecchetto and W. Heidrich, "Probabilistic inverse dynamics for blood pattern reconstruction," in *Vision, Modeling, and Visualization (2011)*. The Eurographics Association, 2011, pp. 369–376.

[3] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006.

[4] T. Balch, Z. Khan, and M. Veloso, "Automatically tracking and analyzing the behavior of live insect colonies," in *AGENTS '01*. ACM, 2001, pp. 521–528.

[5] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 3457–3464.

[6] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[7] A. Milan, K. Schindler, and S. Roth, "Detection- and trajectory-level exclusion in multiple object tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[8] A. Roshan Zamir, A. Dehghan, and M. Shah, "Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs," in *Computer Vision – ECCV 2012*, ser. Lecture Notes in Computer Science, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, pp. 343–356. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33709-3_25

[9] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems. Revised reprint.*, revised reprint ed. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2013.

[10] H. Kuhn, "The Hungarian Method for the assignment problem," *Nav. Res. Logist.*, vol. 2, no. 1-2, pp. 83–97, 1955.

[11] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *SIAM J. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.

[12] F. Bourgeois and J.-C. Lassalle, "An extension of the Munkres algorithm for the assignment problem to rectangular matrices," *Commun. ACM*, vol. 14, pp. 802–804, December 1971. [Online]. Available: http://doi.acm.org/10.1145/362919.362945

[13] D. Dasgupta, G. Hernandez, D. Garrett, P. K. Vejandla, A. Kaushal, R. Yerneni, and J. Simien, "A comparison of multiobjective evolutionary algorithms with informed initialization and Kuhn-Munkres algorithm for the sailor assignment problem," in *GECCO '08*. ACM, 2008, pp. 2129–2134.

[14] J. Bijsterbosch and A. Volgenant, "Solving the rectangular assignment problem and applications," *Ann. Oper. Res.*, vol. 181, pp. 443–462, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10479-010-0757-3

[15] A. B. Poore and S. Gadaleta, "Some assignment problems arising from multiple target tracking," *Math. Comput. Model.*, vol. 43, no. 9-10, pp. 1074–1091, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S089571770500525X

[16] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Autom. Control*, vol. 24, no. 6, pp. 843–854, Dec. 1979.

[17] C. Veenman, M. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 54–72, Jan 2001.

[18] M. Betke, D. E. Hirsh, A. Bagchi, N. I. Hristov, N. C. Makris, and T. H. Kunz, "Tracking large variable numbers of objects in clutter." in *CVPR '07*. IEEE, 2007, pp. 1–8.

[19] Z. Khan, T. Balch, and F. Dellaert, "Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model," in *IROS 2003*, vol. 1. IEEE, Oct. 2003, pp. 254–259.

[20] ——, "MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1805–1918, 2005.

[21] ——, "Multitarget tracking with split and merged measurements," in *CVPR '05*, vol. 1. IEEE, 2005, pp. 605–610.

[22] D. Grover, J. Tower, and S. Tavaré, "O fly, where art thou?" *J. Roy. Soc. Interface*, vol. 5, no. 27, pp. 1181–1191, Oct. 2008. [Online]. Available: http://rsif.royalsocietypublishing.org/content/5/27/1181.abstract

[23] A. D. Straw, K. Branson, T. R. Neumann, and M. H. Dickinson, "Multi-camera real-time three-dimensional tracking of multiple flying animals," *J. Roy. Soc. Interface*, vol. 8, no. 56, pp. 395–409, Mar. 2011. [Online]. Available: http://rsif.royalsocietypublishing.org/content/8/56/395.abstract

[24] Y. Liu, H. Li, and Y. Chen, "Automatic tracking of a large number of moving targets in 3d," in *Computer Vision – ECCV 2012*, ser. Lecture Notes in Computer Science, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, vol. 7575, pp. 730–742. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33765-9_52

[25] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2004. [Online]. Available: http://books.google.com/books?id=si3R3Pfa98QC

[26] A. Tyagi, G. Potamianos, J. Davis, and S. Chu, "Fusion of multiple camera views for kernel-based 3d tracking," in *Motion and Video Computing, 2007. WMVC '07. IEEE Workshop on*, Feb 2007, p. 1.

[27] A. Tyagi, M. Keck, J. Davis, and G. Potamianos, "Kernel-based 3d tracking," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–8.

[28] R. Eshel and Y. Moses, "Homography based multiple camera detection and tracking of people in a dense crowd," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[29] B. Kausler, M. Schiegg, B. Andres, M. Lindner, U. Koethe, H. Leitte, J. Wittbrodt, L. Hufnagel, and F. Hamprecht, "A discrete chain graph model for 3d+t cell tracking with high misdetection robustness," in *Computer Vision – ECCV 2012*, ser. Lecture Notes in Computer Science, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, vol. 7574, pp. 144–157. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33712-3_11

[30] H. Park, T. Shiratori, I. Matthews, and Y. Sheikh, "3d reconstruction of a moving point from a series of 2d projections," *Computer Vision–ECCV 2010*, pp. 158–171, 2010.

[31] L. A. Zarrabeitia, F. Z. Qureshi, and D. A. Aruliah, "Droplet tracking from unsyncrhonized cameras," in *Proc. Second International Conference on Pattern Analysis and Applications (ICPRAM 2013)*, Barcelona, Spain, February 2013, p. 8pp.

[32] L. A. Zarrabeitia, D. A. Aruliah, and F. Z. Qureshi, "Extraction of blood droplet flight trajectories from videos for forensic analysis," in *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, vol. 2. SciTePress, 2012, pp. 142–153.

[33] A. Andriyenko, K. Schindler, and S. Roth, "Discrete-continuous optimization for multi-target tracking," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1926–1933.

[34] R. Murray, "Computational and laboratory investigations of a model of blood droplet flight for forensic analysis," Master's thesis, University of Ontario Institute of Technology, 2012.

[35] Z. Wu, N. I. Hristov, T. Kunz, and M. Betke, "Tracking-reconstruction or reconstruction-tracking? comparison of two multiple hypothesis tracking approaches to interpret 3d object motion from several camera views," in *Motion and Video Computing, 2009. WMVC '09. Workshop on*, Dec 2009, pp. 1–8.

[36] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 03 1964. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177703732

[37] D. Hestenes, *New Foundations for Classical Mechanics*. Kluwer Academic Publishers, 1999.

[38] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, Nov 2000.

Luis A. Zarrabeitia received his BSc. in Computer Science (2005) and his MSc. in Applied Mathematics (2009) from the University of Havana. He has worked at the University of Havana (lecturer, computer systems manager), at the Institute of Cybernetics, Mathematics and Physics in Havana (researcher with the CAGD group) and at Nascent Digital (web and mobile application developer). He is currently a Ph.D. candidate in Computer Science with the Visual Computing lab at UOIT. His research interests include computer vision and high-performance computing.

Faisal Z. Qureshi received M.Sc. and Ph.D. degrees in Computer Science from the University of Toronto, Toronto, Canada, in 2000 and 2007, respectively. He is an Associate Professor of Computer Science and the (founding) director of the Visual Computing lab. He joined the UOIT in 2008 from Autodesk Canada Co. in Toronto, where he was a Software Developer. His research interests include sensor networks, computer vision, and computer graphics. He has also published papers in space robotics. He is a member of the IEEE, the ACM and the CIPPRS.

Dhavide A. Aruliah received an MSc. in Mathematics from Simon Fraser University (Burnaby, Canada) in 1996 and a PhD. in Computer Science from the University of British Columbia (Vancouver, Canada) in 2001. He held a postdoctoral fellowship at the Fields Institute for Research in the Mathematical Sciences (Toronto, Canada) from 2001 to 2002, followed by another at Western University (London, Canada) from 2002-2004. Since 2004, he has been on faculty at the University of Ontario Institute of Technology (Oshawa, Canada) in the Faculty of Science. His primary research interests are computational inverse problems and high-performance scientific computing.