Pattern Recognition 52 (2016) 384–396

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Automatic 3D tracking system for large swarm of moving objects

Ye Liu [a,b], Shuohong Wang [b], Yan Qiu Chen [b,*]

[a] School of Automation, Nanjing University of Posts and Telecommunications, Nanjing, China
[b] School of Computer Science, Fudan University, Shanghai, China

ARTICLE INFO

Article history:
Received 28 March 2014
Received in revised form
12 October 2015
Accepted 14 November 2015
Available online 22 November 2015

Keywords:
3D tracking
Target swarm
Particle filter
3D reconstruction
Multiple hypothesis tracking

ABSTRACT

Natural systems such as bird flocks, fish schools and insect swarms consist of a large group of moving individuals. For many years, scientists have been interested in the complex 3D motion patterns and dynamics they exhibit, trying to discover enlightening rules and causes behind them. Unfortunately, the lack of effective techniques to accurately measure the real 3D trajectories of the individuals had limited the quantitative study on these systems. We propose in this paper an automatic tracking system which is able to track a large number of tiny animals in a 3D volume with multiple cameras. Most visual details of such targets are lost in the captured images because of limited image resolution, and the remainder can be easily corrupted due to frequent occlusion or motion blur, which makes it difficult to establish cross-view and cross-frame correspondences. We formulate the problem as a repeated process of hypothesis generation and verification. Hypotheses are generated when cross-view matching ambiguities occur and are verified at an efficient 3D tracking stage where targets are modeled in 3D space and weak yet existing visual information from multi-view video streams are furthest collected. The whole system is fully automatic in dealing with variable number of targets and robust against detection and matching errors.

1. Introduction

Many of us have been fascinated at least once by the complex motion patterns exhibited by flocking birds, schooling fish and swarming insects. Such complex dynamic collective behaviors of animal groups have attracted significant scientific attention [1,2]. Quantitative study of such patterns and dynamics will advance our knowledge about these behaviors as well as the mechanisms that produce such behaviors [3,4] whose value is not limited to biology but may also be helpful in designing multi-agent robots, i.e. multiple robots (typically Micro Aerial Vehicles) work cooperatively to accomplish certain tasks. The arguably most informative way to quantitative analysis is via the three dimensional (3D) motion trajectory of each individual. Such data however had until recently been scarce due to technological difficulties. GPS devices are helpful with pigeon groups [2], but it is infeasible to attach them to hundreds of tiny animals such as insects.

Recent advances in high-speed and high-resolution cameras have made it possible to capture detailed video streams of such dynamic scene from multiple perspective angles. But automatically recovering the 3D trajectories from obtained 2D video streams is still challenging. In order to recover the 3D trajectories of the targets, multi-view observation data should be integrated in an effective way. Even when synchronized cameras are fixed and calibrated, we cannot easily establish correspondences between two views, because photometric consistency becomes ineffective when dealing with targets that share similar appearance, so matching ambiguity frequently exists (Fig. 1). What makes the problem even more complicated is that occlusion happens among targets frequently, which is difficult for existing 2D tracking methods to overcome. The tracker has to handle frequent distractions from other targets and noise contained in the images. Frequent occlusion can also prevent target detector from distinguishing the targets correctly from images.

Existing methods that solve similar problems usually adopt a detection and association framework: detections from a detector are associated across different views and through consecutive time steps to generate 3D trajectories. The problem with such framework is that it highly depends on the detection results thus detection errors may cause performance degradation, and that the information contained in image data is simply ignored after detection thus is not made full use of.

We propose in this paper a tracking system with a significantly different framework: ambiguities in across-view matching and target detection are resolved by an effective 3D tracker, which is able to estimate the 3D state of a target by integrating multiple sources of information from multiple views. Fig. 1 illustrates the ambiguity resolving strategy. All the ambiguous matching candidates can be triangulated forming a 3D point set, then a 3D tracker

* Corresponding author.
E-mail addresses: yeliu@njupt.edu.cn (Y. Liu), sh_wang@fudan.edu.cn (S. Wang), chenyq@fudan.edu.cn (Y.Q. Chen).

http://dx.doi.org/10.1016/j.patcog.2015.11.014
0031-3203/

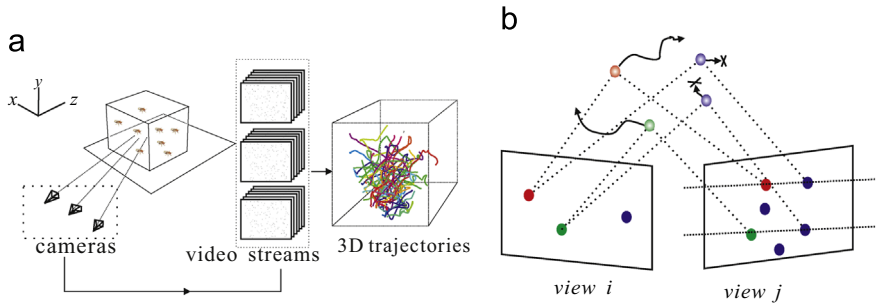

Fig. 1. (a) Experimental setup. A large group of flying animals move in a 3D volume and are imaged by 3 video cameras. The tracking system recovers the 3D trajectories of the targets from the video streams. (b) Figure that illustrates the motivation. Targets that are falsely matched are expected to be tracking inconsistent.

is initialized trying to track each point or in another word checking its tracking consistency. The 3D tracker is expected to be able to stably track real targets and distinguish those phantom ones by checking their tracking consistencies.

The performance of the system is evaluated systematically on both synthesized and real-world data. Our system is capable of generating high-quality 3D trajectories which is helpful for biologists to analyse the behavior of flying animals. The rest of this paper is organized as follows: Section 2 reviews existing work. Section 3 gives an overall flowchart of the proposed tracking system. Section 4 gives the details of how hypotheses are generated. And Section 5 gives a detailed description of the proposed 3D tracker. Sections 6 and 7 introduce backward tracking and duplication removal respectively. Section 8 discusses the experiment results, and finally the conclusion is drawn in Section 9.

2. Related work

Particle filter (PF) has been successfully applied and extended to various tracking problems ever since it was first introduced in visual tracking [5], because of its simplicity and the capability to handle nonlinearity and non-Gaussianity. Many researchers have extended particle filter to multi-target tracking [6–9]. As for single-view multi-target tracking, numerous approaches have been proposed including multiple hypothesis tracking (MHT) [10], JPDAF [11] and greedy assignment [12]. Okuma et al. [7] proposed a particle filter based tracking method for tracking hokey players using color histogram as observation model, when a player newly enters the scene, it is detected and a tracker is started for it. A system for tracking multiple players in sports videos was proposed in [13]. A data association strategy for cell tracking was proposed in [14]. The mutual interaction among multiple humans was modeled to guide tracking in [15]. Khan et al. [9] proposed a particle filter based method to track multiple targets that frequently interact with each other. They used MCMC to sample the resulted high-dimensional state space. For 3D object tracking, a framework that shares similarities to ours was proposed in [16], which tracked multiple line segments in the scene.

Efforts have been made towards 3D tracking of multiple humans with multiple cameras [17–19]. Human bodies are highly deformable and their motion is complex but usually restricted on the ground plane, the targets we track are tiny with little visual details in captured images, the population size is much larger and they are likely to freely fly to any area of a 3D volume. Compared with human tracking, occlusion in our problem happens much more frequently but each occlusion event lasts for shorter period.

Several vision based systems have been developed for biological research. Ref. [1] captured images of starling flock, and reconstructed their 3D positions. The flight kinematics of several wild mosquitoes are tracked in [20]. Ref. [21] tracked a relatively fewer number of fruit flies using multiple cameras. The density of targets in these systems is relatively low (most of the target numbers are below 20), so across-view matching ambiguities can be overcome using the epipolar constraint, and occlusion among targets is not as severe as in our problem.

Some previous work have attempted to track similar number of similar tiny targets in 3D. Du et al. [22] tracked particles using simply the nearest-neighbor strategy which first produce 2D trajectory segments, and then these segments are matched across different views using the epipolar constraint. Although some good stereo matching results were reported, trajectories were broken into many segments which is undesirable for many applications. Zou et al. [23] proposed an off-line tracking algorithm which sought to minimize a global energy function via dynamic programming. Wu et al. [24] proposed a method that linked the trajectory segments generated from 2D tracking. Their methods involve multiple high-cost linear assignment steps. The one to one assumption of linear assignment makes the trajectories prone to breaking up because if a detected object has been assigned to one tracker it cannot be assigned to another tracker which is invalid when occlusion occurs. This constraint was relaxed in [25]. All these methods highly depend on detection results.

3. Overview of the proposed system

A flowchart of the tracking system is shown in Fig. 2. In forward tracking stage, the system maintains a set of 3D trackers, which are initialized for tracking the hypothesized targets. In each time step, some of the trackers that are found inconsistent are terminated and new trackers for the newly generated hypotheses are added to the tracker set. New hypotheses are generated by establishing stereo correspondences for the detections that are not occupied by existing trackers. After all the frames have been processed in the first loop, a backward tracking procedure is carried out to complete the trajectories with a delayed initialization or fragmentations. Finally, after remedying very few problematic trajectories, high-quality 3D trajectories can be generated.

4. Generation of hypothesized trackers

4.1. Target detection

Although the scene is cluttered with a large number of targets, each moving target stays in an area of image for a very short period of time. Thus we can model the background using a temporal median filter. Background of view i at t is computed as the median image of images $t-p$ to $t+p$. Subtracting the background yields a residual image

$$R_t^i(x,y) = |I_t^i(x,y) - median(I_{t-p}^i(x,y),\ldots,I_{t+p}^i(x,y))|. \quad (1)$$

And in the first and last p frames, we use the median image computed in frame $p+1$ and $T-p$ respectively. A pixel (x,y) of R_t^i with a larger value is more likely to be a foreground pixel, thus thresholding the residual image (the threshold is termed as *thre*) yields a binary image which contains many connected components. The barycenters of these connected components are computed as potential positions of the targets. Fig. 3 gives an example of detection results.

4.2. Bi-epipolar stereo matching

The detections responses of view i at t are denoted by a point set E_i^t. And $E_i^t = Es_i^t \cup Eu_i^t$, each point e_k in E_i^t is either assigned to a currently active tracker (included in set Es_i^t) or unassigned (in Eu_i^t). Assignment is done simply by checking if a tracker's estimated 3D position is projected in the connected component of a detection. New trackers are generated only from unassigned detections.

Even with the epipolar constraint, a detection in view i may find several candidates in view j. We introduce the epipolar constraint in previous time step to help reduce ambiguities (Fig. 4), we name this bi-epipolar constraint. The across-time correspondence is established with template matching [26] in a neighborhood of $\mu \times \mu$. With the bi-epipolar constraint, the number of trackers maintained can be reduced by at least 50% which saves computational resources. We set the epipolar matching threshold (termed as ψ) (i.e. the distance to the epipolar line) to 4 pixels.

Both the detection method and stereo matching are simple and work well if no occlusion happens, which is sufficient for our framework. A target may fail to be initialized at current time step due to occlusion, but it can still have chance to be initialized at later time steps when occlusion disappears, as it is highly unlikely that an occlusion lasts for the whole duration of the recording. The bi-epipolar constrain also provides a two-time-step initialization to the 3D tracker which will be introduced later. The following procedure summarizes the process of new tracker generation:

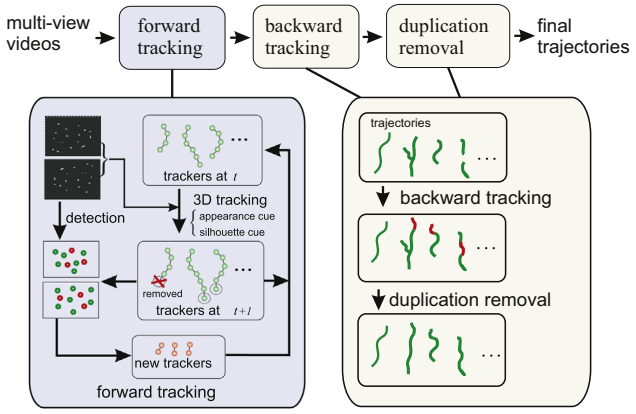

Fig. 2. Flowchart of the tracking system. The top shows the three modules of the tracking system. And the bottom is the detailed procedure of the modules.

Algorithm 1. The procedure of new tracker generation.

1. Trackers estimate new positions (Section 5).
2. Do detection in all the views (generate E_i^t).
3. Assign the detections to trackers (determine Es_i^t and Eu_i^t).

for each e_k^i in unassigned point set Eu_i^t **do**

4. Search in Eu_j^t with epipolar for points: $C = \{e_1^j, e_2^j, \ldots, e_L^j\}$.

if C is not empty, **then**

5. Find a position at $t-1$ using template matching as $tm(e_k^i)$.

for each candidate e_l^j in C **do**

6. Find a position at $t-1$ using template matching as $tm(e_l^j)$.

if $tm(e_k^i)$ and $tm(e_l^j)$ are in the areas of unassigned detections and satisfy the epipolar constraint **then**

7. Initialize a new tracker using e_k^i, e_l^j, $tm(e_k^i)$ and $tm(e_l^j)$

end if

end for

end if

end for

5. Effective 3D tracker

5.1. Target state and dynamic model

Different from conventional tracking methods which estimate the 2D positions of a target on images, the proposed 3D tracker directly estimates the 3D positions of a target from multi-view image observations. We adopt a second-order state space $S_t = (X_t, X_{t-1})^T$ in order to better handle non-constant motion, where X_t is the 3D position of the target at t. And a linear dynamic model is adopted:

$$S_t = BS_{t-1} + v_{t-1}, \tag{2}$$

where $v_{t-1} \sim \mathcal{N}(0, \Sigma)$ is Gaussian noise and B is a 6×6 state transition matrix defined as

$$B = \begin{bmatrix} 2I_3 & -I_3 \\ I_3 & 0 \end{bmatrix}, \tag{3}$$

where I_3 is a 3×3 identity matrix.

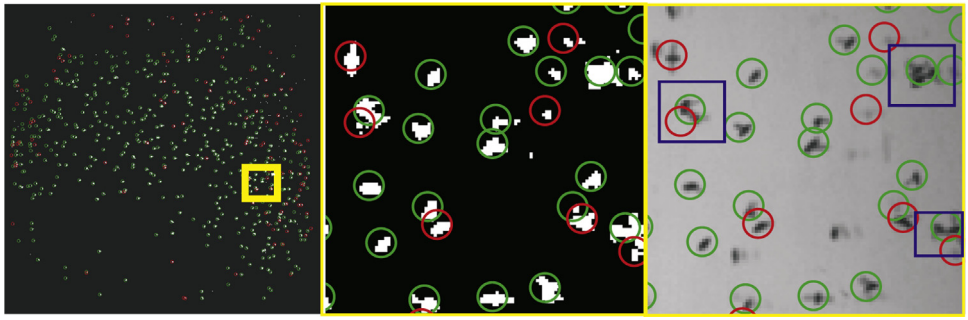

Fig. 3. Detection result of a scene containing about 400 fruit flies. Most of the detection responses (marked with green circles) are assigned to the active trackers. New trackers are initialized for the rest of them (red circles). The detection results are problematic when occlusion happens (blue boxes on the right-most image). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

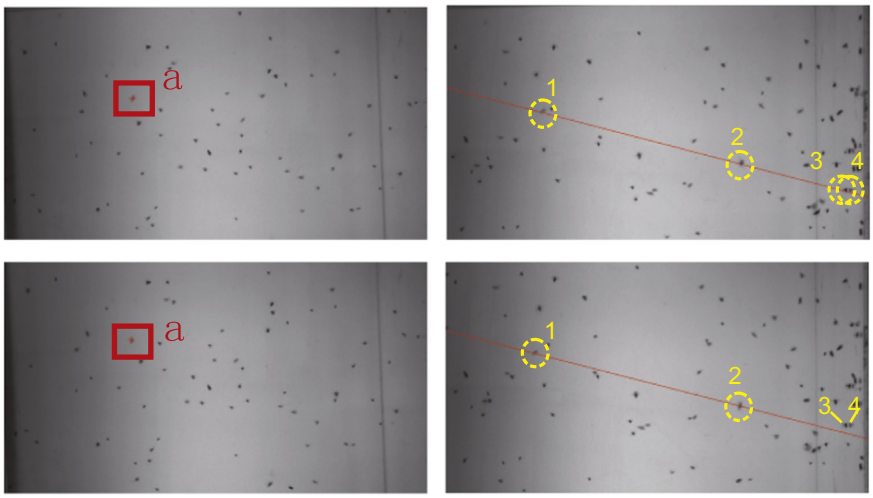

Fig. 4. An example of bi-epipolar constraint. At time t, a target in the first view may find 4 candidate on corresponding epipolar line. The number of candidates is reduced to 2 when another epipolar constraint is applied.

In real 3D space, the metric is physically meaningful, so the kinetic model is more likely to give accurate predictions. Targets which satisfies the bi-epipolar constraint will have two-time-step positions and a tracker for it can be initialized directly.

5.2. Observation model

If the data available at time t is denoted as Z_t, then tracking can be formulated as a problem of estimating the posterior probability $p(S_t|Z_{1:t})$, where $Z_{1:t}$ is the collection of $Z_1, Z_2, \ldots Z_t$. Under the Markov assumption and by Bayes' rule, we can get the well-known equation of Bayesian filtering

$$p(S_t|Z_{1:t}) \propto p(Z_t|S_t) \int p(S_t|S_{t-1}) p(S_{t-1}|Z_{1:t-1})\, dS_{t-1}, \tag{4}$$

where $p(Z_t|S_t)$ is the observation model.

In our problem, $Z_t = \{z_t^i\}_{i=1}^{N_v}$ is the collection of data at t from all the N_v views, and we can establish the relation between 3D and 2D via the camera matrix P_i as $x_t^i = P_i X_t$, thus

$$P(Z_t|S_t) = \prod_{i=1}^{N_v} p(z_t^i|x_t^i). \tag{5}$$

We call $p(z_t^i|x_t^i)$ a 2D observation model. For each 2D observation model, we collect two sources of information: temporal appearance coherency and silhouette consistency. We will illustrate that these two sources are mutually complementary. And finally, the observation model can be written as

$$P(Z_t|S_t) = \prod_{i=1}^{N_v} \underbrace{p_{app}(z_t^i|x_t^i)}_{\text{appearance}} \underbrace{p_{sil}(z_t^i|x_t^i)}_{\text{silhouette}}. \tag{6}$$

5.2.1. Temporal coherency in appearance

Appearance coherency is taken advantage of by nearly every visual tracker. In our problem, there is some characteristic in the appearance of targets: each target may take up only a few pixels which means that most of the visual details are lost during the imaging process. So a majority of the features that have been commonly used in state-of-the-art tracking algorithms are invalid here, for example, color, texture, and key point features. Fortunately, we found simple combination of the pixel values as a feature vector and calculation of the Normalized Cross Correlation (NCC) between different feature vectors as similarity function works pretty well. We think the reason is that further feature extraction on a few pixels causes severe information loss, while the NCC between pixel value vectors can measure the subtle variations caused by pixel-level movements.

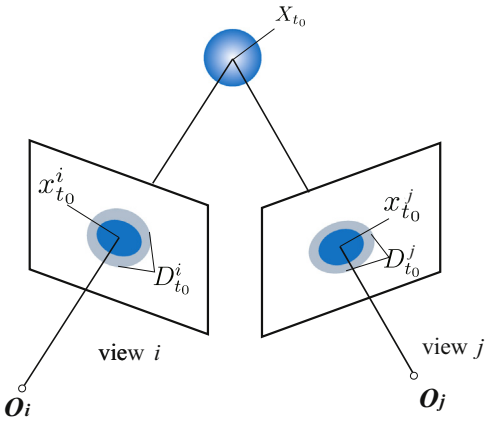

Fig. 5. Reference computation in the appearance cue. At t_0, a disk $D_{t_0}^i$ that is generally larger than the area of the projection of target is selected in each view, and the pixels in the disk are chosen as reference which is compared in later tracking using NCC.

If X and Y are two vectors of dimension n, the NCC score between them is defined as

$$N(X,Y) = \frac{1}{n} \sum_{i=1}^{n} \frac{(X(i)-\overline{X})(Y(i)-\overline{Y})}{\sigma_X \sigma_Y}. \tag{7}$$

where $\overline{X}$ and $\overline{Y}$ are the sample mean of X and Y, σ_X and σ_Y are their standard deviations.

As shown in Fig. 5, consider a target at frame t_0, for image $I_{t_0}^i$ of view i, we select a disk $D_{t_0}^i$ of radius r around the projection $x_{t_0}^i$ as a reference disk. At t, a new disk $D_t{}^i$ of the same size is selected in $I_t{}^i$, then the temporal coherency term in 2D observation model can be computed as

$$p_{app}(z_t^i|x_t^i) \propto \exp\{\alpha N(I_t^i(D_t^i), I_{t_0}^i(D_{t_0}^i))\} \tag{8}$$

5.2.2. Silhouette consistency

The advantage of the temporal appearance coherency cue is that it is capable of grabbing the appearance of a target according to a reference more specifically, while the silhouette consistency on the other hand, can provide universal and unbiased description of the targets, as the performance of the background substraction

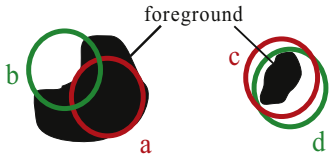

Fig. 6. Soft silhouette constraint. According to the two assumptions, predicted state *a* is more likely to be correct than *b* as it contains more foreground pixels. *c* is better than *d* as the foreground pixels in *c* are closer to its center than those in *d*.

method does not change with time. In our problem, however, it is difficult to obtain perfect segmentation results. So bias may be resulted if constraints that are too strong are added when imperfect silhouettes are given. Therefore we make two soft assumptions:

- An estimated target state whose projections are in the foreground area is more likely to be a correctly estimated one.
- Pixels that are closer to the center of a disk bear higher expectations to be foreground pixels.

These two are better illustrated in Fig. 6. According to the two assumptions, the silhouette consistency term of the 2D observation model can be calculated as

$$p_{sil}(z_t^i|x_t^i) \propto \exp\left\{\beta \sum_{x \in D_t^i} u(x)b(x)\right\}, \tag{9}$$

where $u(x)$ is proportional to a Gaussian kernel that decrease as the distance to the center x_t^i grows, and $b(x)$ is an indicator function that equals one when x is a foreground pixel, and zero otherwise. They are computed as

$$u(x) = \frac{g(|x - x_t^i|)}{\sum_{x \in D_t^i} g(|x - x_t^i|)}, \quad g(x) = \exp\left(\frac{-x^2}{\sigma}\right), \tag{10}$$

and

$$b(x) = \begin{cases} 1 & R_t^i(x) > thre, \\ 0 & \text{otherwise}. \end{cases} \tag{11}$$

5.3. Particle filtering

Instead of assuming a Gaussian distribution as in Kalman filter, particle filter approximates the posterior probability with a set of particles $\{(S_t^n, w_t^n)\}_{n=1\ldots M}$, which can be regarded as multiple hypotheses that are to be tested on observation data. Each particle is associated with a weight w_t^n which is set to $1/M$ initially. New samples are drawn from particles in the previous step using importance sampling [5] and moved independently by the dynamic model, and then they are reweighted as

$$w_t^n \propto p(Z_t^n|\tilde{S}_t^n), \quad \sum_{n=1}^{M} w_t^n = 1. \tag{12}$$

After we have a particle set that approximates $p(S_t|Z_{1:t})$, we simply compute the expectation as

$$E(S_t|Z_{1:t}) = \sum_{n=1}^{M} w_t^n S_t^n. \tag{13}$$

5.4. Tracking under occlusion

In order to test the effectiveness of the two cues, we randomly choose 100 trajectories from the groundtruth of real data, and initialize a tracker with the starting position of each trajectory, and then test if the tracker can keep track of the target. With

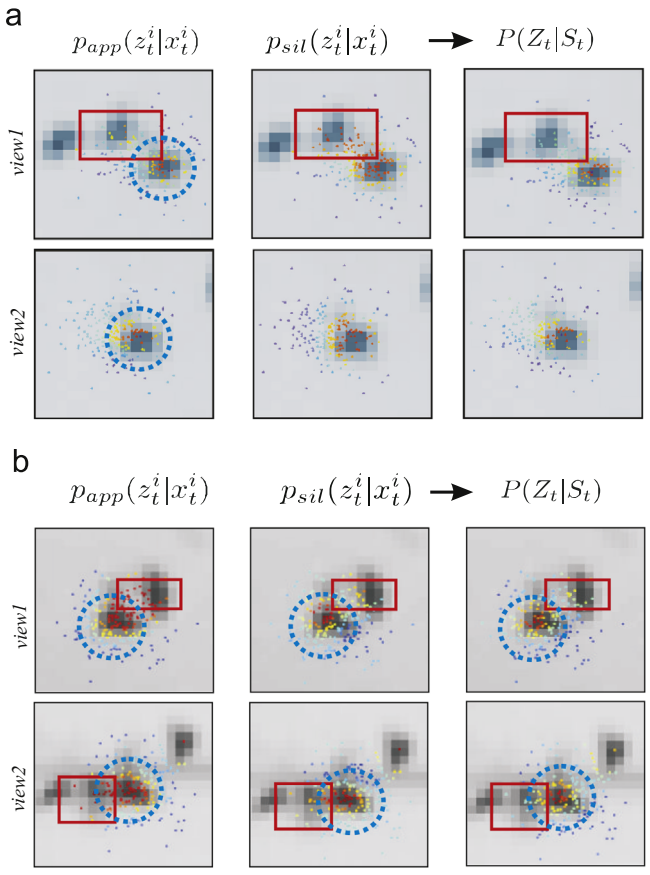

Fig. 7. (a) Tracking under single-view occlusion, the blue dashed circle is the target position which is marked manually. The color change of the particles in the red-boxed area shows how the tracker integrates multi-view information to overcome occlusion. The dots in the red box should be with low weights. (b) Tracking under two-view occlusion, also note the color change of the particles in red-boxed areas. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

appearance cue, 67 of the trackers can keep track of the target all the time and produce complete trajectories. With silhouette cue, the number is 82. And with both cues, the number of completed trajectories is 96. This proves that the combination of the two cues is superior to using only one of them.

Both the appearance coherency and silhouette consistency cues utilized by us can be considered as some kind of soft constrains. When occlusion occurs, we do not expect single 2D observation model $p_{app}(z_t^i|x_t^i)$ or $p_{sil}(z_t^i|x_t^i)$ to give precise predictions, nor will they make significantly biased predictions. What we expect is that if information of multiple cues in multiple views is collected together as a 3D observation model, the tracker is finally able to give correct estimation.

Fig. 7 illustrates how 2D observations are combined to overcome distractions and occlusions. Little colored dots represent particles of the particle filter, red color means a higher weight while blue color means a lower one. We can see that when the observation in view 1 (blue circle) is partially occluded by another target (Fig. 7(a)), the 2D observation models of view 1 are distracted (become red) to some extent, but when all the 2D observation models have been combined, the tracker is hardly affected by distraction (the rightmost image). When partial occlusions happen in two views, both of the 2D observation models in the two views are distracted to some extent (See Fig. 7(b)), but when the 2D observation models of different views are integrated, the tracker is able to overcome these distractions.

5.5. Tracker status assessment and termination

Since the trackers are tasked with checking the correctness of the hypotheses, they should be able to give alarms when tracking consistency is no longer satisfied. We assess the tracking consistency by calculating the number of effective particles. A particle is assumed to be ineffective if the silhouette consistency in one of the views is invalid (i.e. log $p_{sil}(z_t^i|x_t^i)$ is below a threshold (termed as τ)). If the number of effective particles is below a threshold κ for 5 consecutive frames, the tracker is terminated.

After all the frames have been processed, several trackers may have been initialized for a detection response by the bi-epipolar matching, and the tracker with the longest trajectory is kept. If the length of the longest trajectory is shorter than 20, we assume the detection to be false positives and discard all of the initialized trackers. Detection false positive may cause the initialization of a few false hypotheses (3D trackers), but they will be terminated timely due to tracking inconsistency. Thus the system is robust against detection failures.

6. Backward tracking

Each true target hypothesis is generated at occlusion-free instant, so it is likely that the target is initialized in the middle of its trajectory. Backward tracking can complete these trajectories. Also, trajectory segments are likely to be linked by backward tracking.

Before backward tracking is started, the system also maintains a set of trackers from previous stages. From the last time step T to 0, the system finds those trackers that are "backtrackable", reinitializes them and extends them backward. A tracker is "backtrackable" at t if it is started at t. The tracking process is the same with that in forward tracking, and at each time step, tracker status are checked in the same way as in forward tracking and inconsistent trackers are terminated.

7. Duplication removal

Most of the trajectories obtained so far are correct, but a few of them may have the duplication problem: one target may be tracked by more than one tracker in a sufficiently long period of time. This problem may be caused by the following two kinds of events (as shown in Fig. 8):

- In forward tracking, multiple trackers are initialized for the same target due to duplicated or false detection. These trajectories should be handled so that one merged trajectory is kept, one possible solution is shown in Fig. 8(a).
- In backward tracking, a posterior trajectory is extended and partially covers an anterior one. These trajectories should also be handled so that one merged trajectory is kept, one possible solution is shown in Fig. 8(b).

Most of the trajectory duplications are simply caused by one of the two events. But in a few rare cases, these two kinds of events may be mixed (Fig. 9(a)) making it a complicated problem to remove the duplications. So we propose a unified graph based algorithm to deal with both simple and complicated cases.

7.1. Duplication detection

If two trajectories share an overlap longer than 10 frames, we consider them to be duplicated. We can construct a graph with all the trajectories as its nodes, if two trajectories share a more than 10 time-step overlap, an edge is added between the two nodes. We deal with each connected component of the graph containing more than one node (trajectory) separately.

For a connected component which contains a set of trajectories, a directed graph can be constructed like this: all 3D positions of these trajectories are added as graph nodes, the nodes that are contained in overlaps are merged (see Fig. 9(a)). The number of effective particles is recorded in the tracking process that generates the nodes. And if a node is merged from more than two nodes, the average number of effective particles is computed and kept. Then the problem becomes splitting the merged graph into trajectories with duplications removed.

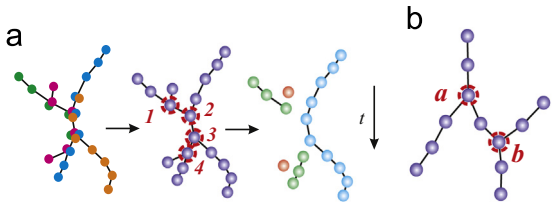

Fig. 9. (a) Graph splitting by searching the junction without dependence. (b) The case of all the junctions are mutually dependent.

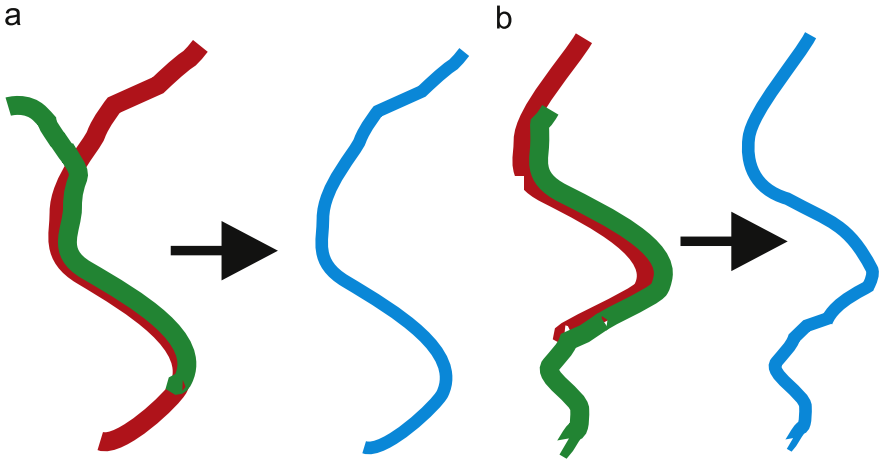

Fig. 8. The two events that may cause duplications. (a) Multiple trackers are initialized for one target. (b) A trajectory is prolonged in backward tracking and it partially covers an anterior one.

7.2. Ambiguity resolving

Ambiguities in splitting the graph are with those nodes whose in-degree or out-degree is more than one, which we call "junctions". And for each junction, a decision should be made so that only one of its (forward or backward) branches is kept. All the junctions should be found firstly. Then we take into account two things in making the decisions: trajectory length and tracking consistency. This is motivated by the two facts that longer trajectories are more likely to be correct than shorter ones, and that tracking consistency can reflect the quality of the tracking process. For each branch j of a junction i, the following score is computed:

$$S(j) = \eta\varphi(len(j)) + (1-\eta)Tc(j) \tag{14}$$

where

$$\varphi(x) = \begin{cases} ax+b & x < \xi, \\ a\xi+b & x >= \xi. \end{cases} \tag{15}$$

and

$$Tc(j) = \sum_{t_j-3<t<t_j+3} N_{eff}(t). \tag{16}$$

The first term $\eta\varphi(len(j))$ is determined by the branch length: it increases with branch length when branch length is below ξ (we set 20 in the experiments). $\eta\varphi(len(j))$ is fixed when length is above ξ. The second term is the sum of the number of effective particles (the same with that in Section 5.5) of the time steps near the junction, which records the tracking consistency of a trajectory. After the scores of branches have been computed, the branch with the highest score is kept.

However, the length of a branch of junction cannot be determined if there is another junction on the branch. We say junction A depends on junction B if B is on one of A's branches. So firstly, the dependencies of the junctions are computed and then a junction that does not depend on others is searched. If such a junction is found, then the graph is split. This procedure is repeated until the graph is split at all the junctions. For example in Fig. 9(a), we have junctions $1\rightarrow 2, 4\rightarrow 3$, and junctions 1 and 4 are split in the first iteration, and then 2 and 3 are split.

It is possible that junctions are mutually dependent then the above procedure will fail. Such an example is shown in Fig. 9(b). In this case, we find and remove the longest trajectory from the graph without considering the tracking consistency, then the rest of the graph is split. In fact, this is a very rare case that can happen only theoretically, as we did not find it in all the experiments. Finally, the duplication removal procedure is summarized as follows:

Algorithm 2. The procedure of duplication removal.

1. Construct a directed graph for the trajectories that share duplication.
2. Find all the junctions $J = \{J_i\}$

while J is not empty **do**
3. Compute the dependencies.
if A junction J_k that does not depend on others can be found in J, **then**
4. Split the graph at J_k,
5. Remove J_k from J.
else
6. Find and remove the longest trajectory from the graph,
7. Update J and go to STEP 2.
end if
end while

7.2.1. Failure cases

However, there are some rare cases that the above strategy is not able to deal with. If two targets collide with each other the trajectories may change abruptly which will probably cause mistakes. Also, our method cannot deal with the case when one target clings to another for more than 10 time steps.

8. Experiments

8.1. Metrics and methods for evaluation

We use both simulated target swarm and real fruit flies (*Drosophila melanogaster*) in our experiments to systematically evaluate the performance of the proposed system and to compare it to state-of-the-art methods.

The ground truth for real-world target swarms is unfortunately difficult to obtain even by manually analysing the video streams. Using simulated targets therefore offers a significant advantage, i.e. ground truth is known in advance so that the correctness of trajectories can be checked conveniently. Another merit is that parameters such as the number of targets is tunable, which makes it possible to carry out multiple experiments for systematical evaluation and comparison. The parameters of the algorithm in the experiments are listed in Table 1.

Generally, these parameters are set empirically with a subset of data. For example, when determining *thre*, we select a subset of the images and check the detection results. And we determine the matching window size μ by checking the distribution of the ground truth movements between two consecutive frames. The parameters of the 3D tracker are tuned by selecting a subset of correctly initialized trackers and test the performance of different parameters. And the parameters of the duplication removal are set with a subset of the duplication cases. Although these parameters are tuned empirically without searching over the entire parameter space, they work well in the experiments.

As discussed above, it is generally difficult to obtain ground truth of 3D trajectories for real-world target swarms, it becomes less difficult when mostly correct trajectories are given (as those generated by proposed method), and the task becomes manually checking the correctness of the trajectories and then repairing a small quantity of problematic trajectories. More details of this check-and-repair strategy will be discussed later.

We adopt several performance metrics that are computed automatically by a procedure. For each trajectory T of the ground truth, the procedure tries to find a trajectory T' from the recovered trajectories which share the longest overlap $O(T, T')$ with T. Two trajectories share an overlap when both the distances between their projections in two views are less than 10 pixels. We say a

Table 1
The parameters of the algorithm in the experiments.

Detection and matching				3D tracker							Duplication removal			
thre	p	μ	ψ	α	β	σ	τ	κ	M	r	a	b	ξ	η
10	4	20	4	20	30	1	0.2	20	200	5	80	−800	20	0.5

ground truth trajectory is completely reconstructed (*completed*) when $|T|-|O(T,T')|<10$. Also, we count the number of trajectories that have been recovered more than 80% (i.e. $|O(T,T')|>0.8|T|$), and those that have been recovered between 20 and 80%. The number of ID switches (*IDS*) and fragmentation (*FRAG*) are also counted from recovered trajectories' perspective, for each recovered trajectory $T'=\{T'_t\}$, we assign an ID of nearest ground truth trajectory to each element of T'. And ID switches can then be automatically detected. And if T' ends up at the middle of a ground truth trajectory (more than 10 time steps from the end), we consider it to be fragmented.

We evaluate the performance of proposed method with methods [25,24,23] that address problems similar to ours. Both [25,23] used two cameras in their experiments, and there is no special mechanisms in the two methods to deal with more than two cameras. The method [23] takes 2D detection responses as input and reconstructs 3D trajectories in a global correspondence selection (GCS) framework. Methods [25,24] share some similarities: they both solve ambiguities in 2D tracking using multi-view information. The difference is that [24] generates 2D trajectory segment firstly, so the ambiguities can be reduced to some extend.

8.2. Simulated target swarms

The arena for simulated targets is a cube whose edge length is 1000 units. At initialization, N targets are randomly generated in the cube: (x,y,z) are generated from three uniform distribution $U(0,1000)$ respectively. Their velocities are of random magnitude and random directions: $v=(v_x,v_y,v_z)$ are generated from three uniform distribution $U(-0.002,0.002)$ respectively. These targets interact with each other via a force field analogous to that in molecular mechanics: attraction force exists between two targets when the distance between them is greater than δ; While repulsive force exists when the distance is less than δ. δ is set to 250 in the experiments. At the same time, each face of the box exerts inward repulsive force to targets that are in its vicinity, which tries to keep the targets inside the cube (see Fig. 10). The force at t determines the acceleration: $a^t=(a_x^t,a_y^t,a_z^t)$. The velocity is updated with the acceleration: $v^t=v^{t-1}+a^t$. The position of the target at t is updated with velocity of $t-1$: $(x^t,y^t,z^t)=v^{t-1}+(x^{t-1},y^{t-1},z^{t-1})$. Targets that escape from the cube are eliminated and an equal number of targets are then initialized inside the cube. Finally, short trajectories (less than 30) are deleted so the number of targets in each frame is not strictly the same.

The targets are rendered using OpenGL with each target being graphically modeled as a ball of fixed radius. Virtual cameras are set to capture videos of the moving targets. We generate two-view data and three-view data respectively. A rendered sample frame and some of the trajectories are shown in Fig. 10.

We simulated several cases with various numbers of targets (N from 50 to 290) to test the performances under different degrees of occlusions. Although the background in simulated data as compared to real data is cleaner, they cannot be regarded as simpler cases, because the appearances of the simulated targets are identical to each other which is a challenge that rarely occurs in real-world targets. So simulated data is a more challenging test of a methods ability in resisting the distractions from other targets.

On the two-view data, the percentages of completed trajectories with respect to target number are plotted in Fig. 11. GCS works well with relatively fewer targets, more than 90% of the trajectories can be completed. As the number of targets grows, ID switches begin to happen frequently, and the percentages drops rapidly. The proposed method, on the other hand, is able to keep a percentage of 90% even with 290 targets. The numbers of fragmentation in the results of GCS are fewer than those in our results, one possible explanation is that the trajectories are over-linked in GCS, which makes the trajectories less likely to be fragmented but

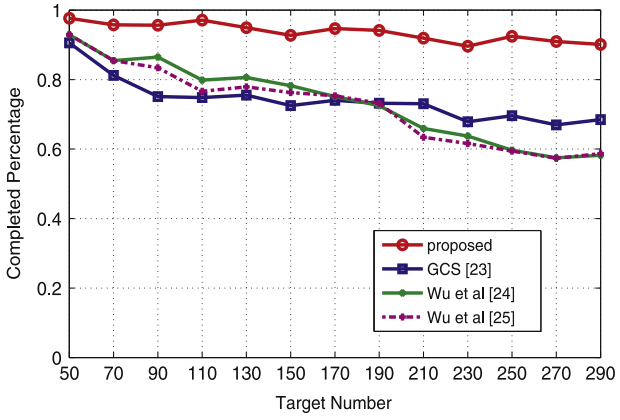

Fig. 11. Percentages of completed trajectories over number of targets for simulated two-view data.

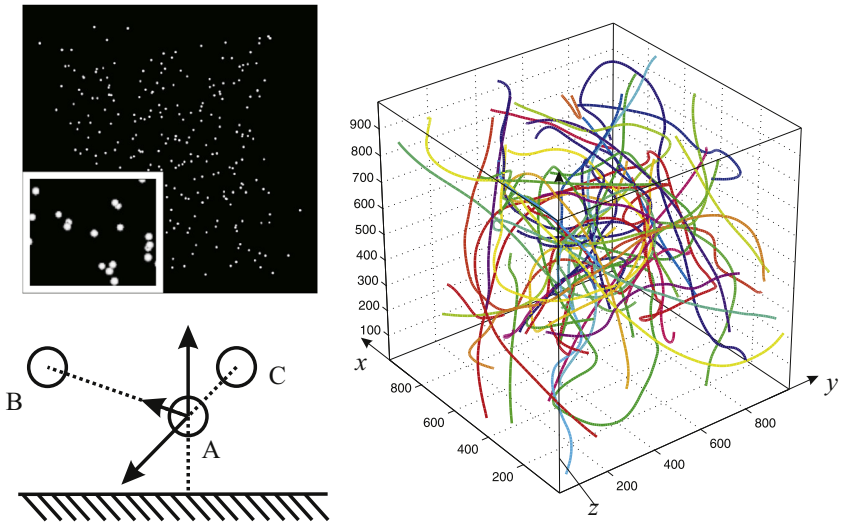

Fig. 10. Top left: example image of simulated data. Bottom left: figure illustrating force field. Forces are exerted to A by target B, C and the face of the box. Right: some of the motion trajectories.

Table 2
Comparison results on simulated two-view data (Part I).

Method	Targets	Trajectories	Completed	80–100%	20–80%	IDS	FRAG
GCS	50	127	115	117	10	15	0
Wu et al. [24]	50	127	118	118	9	7	6
Wu et al. [25]	50	127	118	118	8	6	7
Proposed	50	127	124	127	0	3	1
GCS	70	165	134	144	21	45	0
Wu et al. [24]	70	165	141	151	14	26	12
Wu et al. [25]	70	165	141	146	19	19	17
Proposed	70	165	159	165	0	7	1
GCS	90	229	172	193	36	98	0
Wu et al. [24]	90	229	198	207	22	45	34
Wu et al. [25]	90	229	191	195	17	37	43
Proposed	90	229	221	222	7	14	2
GCS	110	278	208	232	46	133	3
Wu et al. [24]	110	278	222	238	40	57	40
Wu et al. [25]	110	278	213	218	58	53	45
Proposed	110	278	270	270	8	17	2
GCS	130	335	253	275	60	157	2
Wu et al. [24]	130	335	270	287	48	65	74
Wu et al. [25]	130	335	261	269	66	54	88
Proposed	130	335	320	326	8	27	10
GCS	150	400	290	323	77	200	2
Wu et al. [24]	150	400	313	339	61	97	83
Wu et al. [25]	150	400	305	315	85	91	89
Proposed	150	400	370	377	22	51	10
GCS	170	470	348	387	83	247	2
Wu et al. [24]	170	470	353	384	86	105	127
Wu et al. [25]	170	470	354	364	104	92	143
Proposed	170	470	449	459	11	34	6

Table 3
Comparison results on simulated two-view data (Part II).

Method	Targets	Trajectories	Completed	80–100%	20–80%	IDS	FRAG
GCS	190	563	412	464	99	298	7
Wu et al. [24]	190	563	408	439	124	110	143
Wu et al. [25]	190	563	412	424	139	105	152
Proposed	190	563	533	546	13	51	13
GCS	210	631	461	517	114	313	9
Wu et al. [24]	210	631	416	422	209	157	154
Wu et al. [25]	210	631	400	419	212	162	163
Proposed	210	631	585	600	29	79	24
GCS	230	703	477	560	137	379	7
Wu et al. [24]	230	703	448	456	247	178	197
Wu et al. [25]	230	703	433	463	240	183	205
Proposed	230	703	631	648	48	129	28
GCS	250	796	554	649	143	424	5
Wu et al. [24]	250	796	475	480	315	232	211
Wu et al. [25]	250	796	473	495	301	209	233
Proposed	250	796	743	759	33	93	21
GCS	270	896	600	740	155	530	6
Wu et al. [24]	270	896	515	535	361	285	281
Wu et al. [25]	270	896	514	543	353	243	289
Proposed	270	896	817	843	46	124	28
GCS	290	1012	693	814	196	574	15
Wu et al. [24]	290	1012	589	656	354	310	328
Wu et al. [25]	290	1012	594	617	418	339	356
Proposed	290	1012	917	942	66	156	40

more likely to switch their IDs. Methods [25,24] exhibit similar performances as expected. But when dealing with larger number of targets, the performances of both the two methods drop significantly. We also list the performances on the metrics in Tables 2 and 3.

We also give the performances of proposed method and [25] on the three-view data in Fig. 12. The ambiguities can be better solved to some extend in both of the two methods. The role of the duplication removal (DR) stage of our method is also evaluated, we give a plot which shows the performances with and without

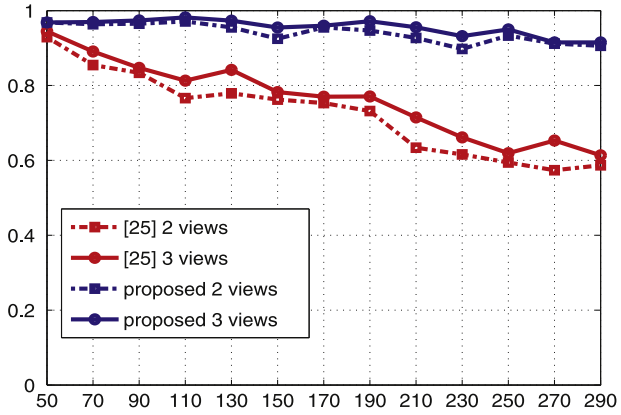

Fig. 12. Percentages of completed trajectories over number of targets for simulated two-view and three-view data.

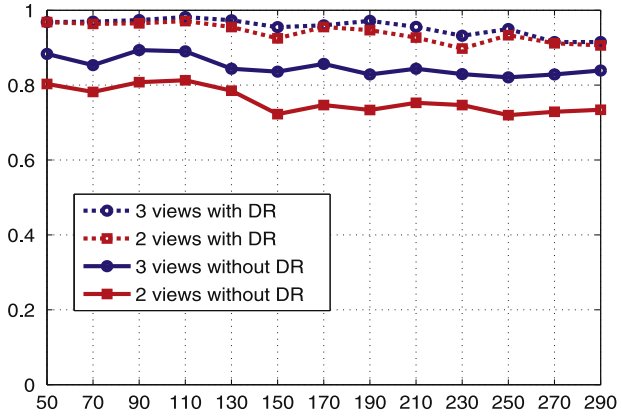

Fig. 13. Percentages of completed trajectories of proposed method with and without duplication removal over number of targets for simulated two-view and three-view data.

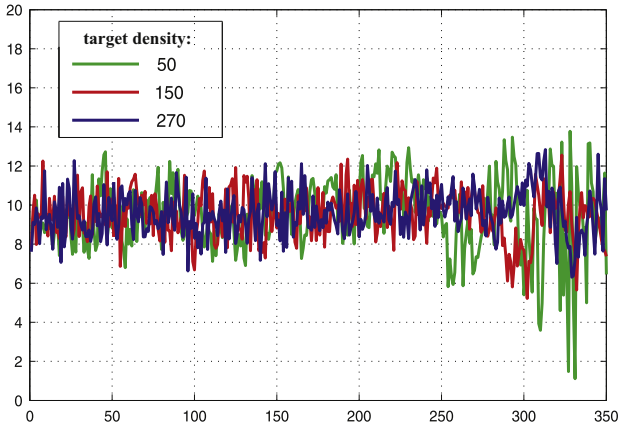

Fig. 14. Error accumulation on 3 datasets with different numbers of targets.

duplication removal on both two-view and three-view data (see Fig. 13).

The drifting problem is common in visual tracking: as time goes on, the tracker gradually deviates from the target being tracked. In order to evaluate if the drifting problem exists in proposed method and to what degree it affects tracking, we give some statistics: for a given t, we compute for each trajectory the distance from estimated position to ground truth at time t_0+t (if t is less than trajectory length), where t_0 is the time step at which the trajectory is started. And we then compute the mean distance at t, thus we can plot a curve whose x-axis is the number of time steps t

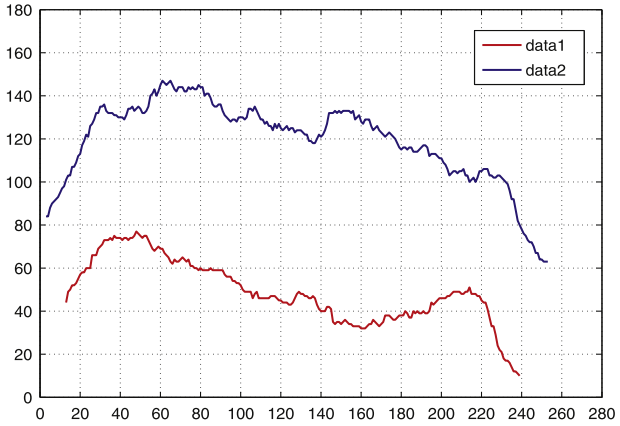

Fig. 15. Numbers of targets that emerge in each time step in two real datasets. The mean number and standard deviation of dataset 1 is 49 and 14.8. And mean and standard deviation of dataset 2 is 119 and 19.1.

from the beginning and y-axis is mean distance as shown in Fig. 14. We can see that as time t grows, the expected mean distance grows very little. The fluctuation becomes greater because less samples (long trajectories) can be found to calculate the mean distance. And the figure also illustrates that larger number of targets does not lead to greater error accumulation.

8.3. *Real-world target swarms*

We use a transparent acrylic box of size 35 cm × 35 cm × 25 cm. Two datasets are collected for testing. Both datasets were captured after two different groups of fruit flies are put into the box, 3 high-speed cameras which have been geometrically calibrated [27] and temporally synchronized are used to capture videos of the animals after they are stimulated. Dataset 2 contains more moving targets because this group of fruit flies are more active and sensitive to stimulation. Dataset 1 is captured using 3 color CCD cameras of 1024×1024 pixels and 120 fps, and dataset 2 is captured using 3 monochrome CMOS cameras of 2040×2048 pixels and 120 fps. We select data from two views to fit the requirements of all the methods for comparison ([25,23] are designed for two cameras) and to make the comparison fair. We also tested the three-view data with proposed method and [25]. For dataset 2 the images are down-sampled to 1020×1024 for computational efficiency. And some dark areas in images of dataset 2 are masked as they may be misleading in manually generating the ground truth. The change of the number of targets emerged is shown in Fig. 15, dataset 2 is more challenging than dataset 1 in that more targets appear simultaneously. We have developed a GUI for manual observation and correction of the trajectories. Each trajectory is projected onto images of two views, and one can observe the tracking consistencies in different views, if 2D trajectories in both of the views are consistent, the trajectory is considered to be a correct 3D trajectory. Fragmented trajectories are either linked or re-initialized manually until all the trajectories are correct. In this way, ground truth is finally generated, some of the trajectories and tracking results are shown in Fig. 16.

The metrics for real-world target swarms are the same as those for simulated experiments. The evaluation results are listed in Tables 4 and 5. The percentages of completed trajectories obtained by proposed method in both of the two datasets are above 90% (97.2% and 95.3% respectively). Although mistakes may exist in the ground truth due to possible ambiguities in manual observation (we only found one such instance in dataset 2 that two targets collide and it was difficult to make a decision afterwards), it will make little impact on the overall performance. The performances

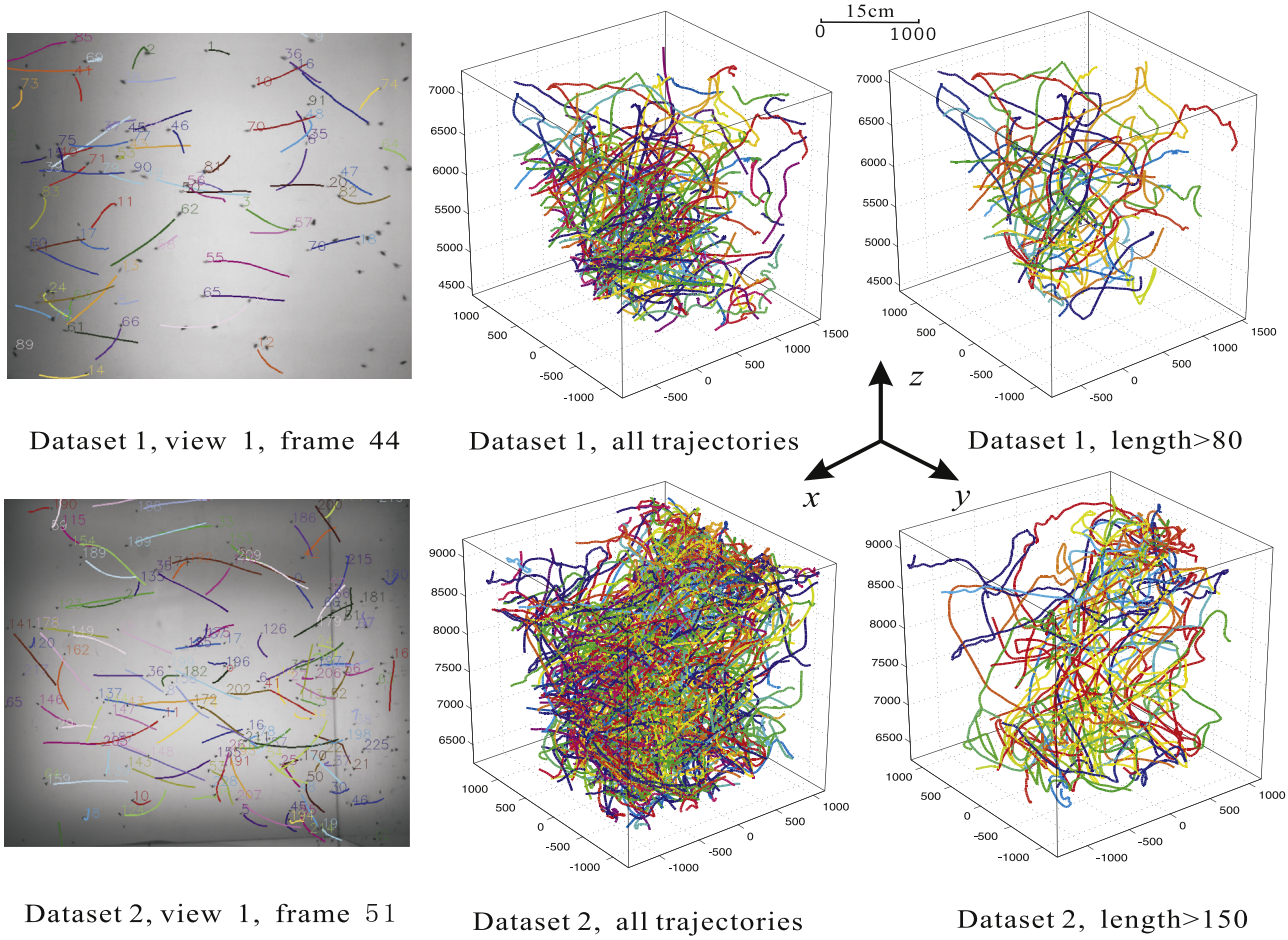

Fig. 16. Tracking results on dataset 1 and dataset 2. Left: trajectories are plotted on image of one view. Middle: all the recovered trajectories of dataset 1 and dataset 2. Right: long trajectories of dataset 1 (length above 80) and dataset 2 (length above 150).

Table 4
Comparison results on fruit fly dataset 1 (two views).

Method	Trajectories	Completed	80–100%	20–80%	IDS	FRAG
GCS	177	163	165	12	36	8
Wu et al. [24]	177	161	160	17	13	24
Wu et al. [25]	177	156	158	19	14	26
Proposed without DR	177	151	149	26	15	31
Proposed	177	172	172	5	6	5

Table 5
Comparison results on fruit fly dataset 2 (two views).

Method	Trajectories	Completed	80–100%	20–80%	IDS	FRAG
GCS	365	306	316	49	138	11
Wu et al. [24]	365	308	316	49	63	74
Wu et al. [25]	365	314	314	27	30	83
Proposed without DR	365	312	319	44	34	69
Proposed	365	348	350	15	17	16

of both methods are higher than those with equivalent number of targets in simulated experiments, which proves that simulated data is more challenging in at least some aspects. In Tables 4 and 5 and We also gives the performances of proposed method without duplication removal (DR). The performances of proposed method and [25] on three-view data are listed in Table 6.

8.4. Computation times

All the experiments are carried out on a PC with Intel Core i5 CPU and 4GB RAM. The proposed method and GCS are implemented in C++ without optimization for parallel processing. The other two methods are implemented with Matlab, the computation times are given only for reference. The rate at which the computation time increases along with the increased number of targets in simulated target swarm reflects the computational complexity of the method. As shown in Fig. 17, the computation time of the proposed method increases linearly with the target number, while the time of GCS climbs nearly exponentially. The proposed method takes 11 minutes for 290 targets while GCS takes more than 48 hours. GCS is faster than our method when the target number is relatively fewer (less than 50).

The proposed method takes 190 s and 315 s for the two fruit fly datasets (two views). And it takes 264 and 405 s for the three-view fruit fly datasets. Each tracker takes fixed amount of time (about

Table 6
Comparison results on fruit fly datasets 1 and 2 (three views).

Method	Trajectories	Completed	80–100%	20–80%	IDS	FRAG
Proposed	177	174	174	3	3	2
Wu et al. [25]	177	163	165	12	7	12
Proposed	365	358	358	7	9	3
Wu et al. [25]	365	336	337	27	17	31

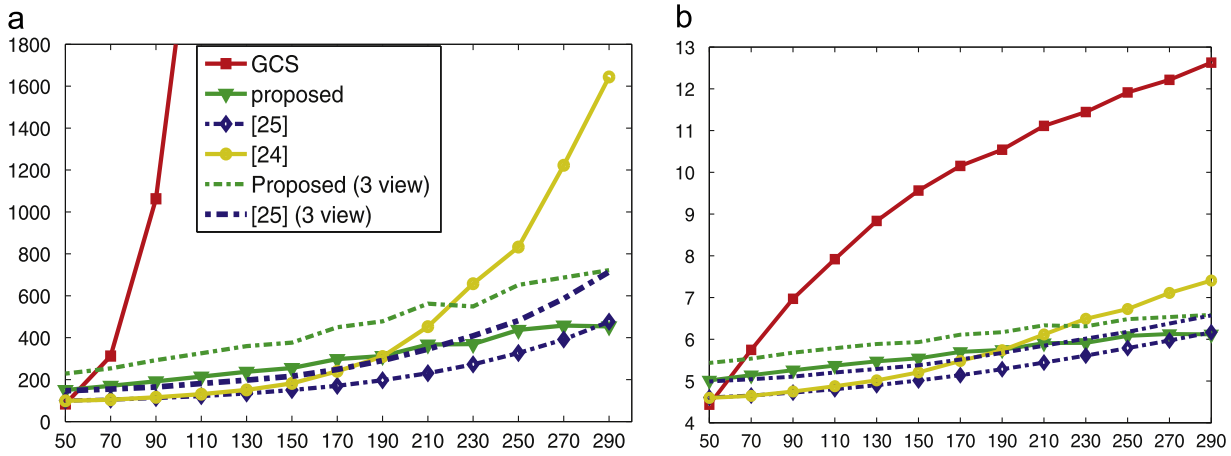

Fig. 17. (a). Computation times of the four methods on simulated target swarms. (b) Computation times in a log space (log(t)) for better visualization.

4.5 ms with 200 particles and radius r set to 5) in estimating a new position. So more than 200 trackers can finish state estimation within one second. Most of the computation time is spend on computing the observation models, it is still possible to speed up the proposed method using multi-threading or GPU, as each tracker runs independently.

9. Conclusion

We have proposed in this paper a tracking system capable of automatically tracking large swarm of moving targets in 3D space. We solve this problem in a hypothesis generation and testing framework. Hypothesis are tested by the proposed 3D tracker working directly in 3D space and information from multiple views is integrated wholistically. Several effective mechanisms have been developed to make the system work fully automatically and robustly.

Conflict of Interest

The authors declared that they have no conflict of interest to this work.

Acknowledgment

The research work presented in this paper is supported by National Natural Science Foundation of China, Grant no. 61175036, and NUPTSF, Grant nos. NY213093 and NY213167.

References

[1] A. Cavagna, A. Cimarelli, I. Giardina, G. Parisi, R. Santagati, F. Stefanini, M. Viale, Scale-free correlations in starling flocks, Proc. Natl. Acad. Sci. 107 (26) (2010) 11865.
[2] M. Nagy, Z. Ákos, D. Biro, T. Vicsek, Hierarchical group dynamics in pigeon flocks, Nature 464 (7290) (2010) 890–893.
[3] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, O. Shochet, Novel type of phase transition in a system of self-driven particles, Phys. Rev. Lett. 75 (6) (1995) 1226–1229.
[4] N. Shimoyama, K. Sugawara, T. Mizuguchi, Y. Hayakawa, M. Sano, Collective motion in a system of motile elements, Phys. Rev. Lett. 76 (20) (1996) 3870–3873.
[5] M. Isard, A. Blake, Condensation—conditional density propagation for visual tracking, Int. J. Comput. Vis. 29 (1) (1998) 5–28.
[6] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, in: European Conference on Computer Vision, Copenhagen, Denmark: Springer, 2002, pp. 661–675.
[7] K. Okuma, A. Taleghani, N. Freitas, J. Little, D. Lowe, A boosted particle filter: multitarget detection and tracking, in: European Conference on Computer Vision, Springer, 2004, pp. 28–39.
[8] C. Hue, J. Le Cadre, P. Perez, Sequential monte carlo methods for multiple target tracking and data fusion, IEEE Trans. Signal Process. 50 (2) (2002) 309–325.
[9] Z. Khan, T. Balch, F. Dellaert, MCMC-based particle filtering for tracking a variable number of interacting targets, IEEE Trans. Pattern Anal. Mach. Intell. 27 (11) (2005) 1805–1819.
[10] D. Reid, An algorithm for tracking multiple targets, IEEE Trans. Autom. Control 24 (6) (1979) 843–854.
[11] T. Fortmann, Y. Bar-Shalom, M. Scheffe, Sonar tracking of multiple targets using joint probabilistic data association, IEEE J. Ocean. Eng. 8 (3) (1983) 173–184.
[12] C. Veenman, M. Reinders, E. Backer, Resolving motion correspondence for densely moving points, IEEE Trans. Pattern Anal. Mach. Intell. 23 (1) (2001) 54–72.
[13] J. Xing, H. Ai, L. Liu, S. Lao, Multiple player tracking in sports video: a dual-mode two-way Bayesian inference approach with progressive observation modeling, IEEE Trans. Image Process. 20 (6) (2011) 1652–1667.
[14] A. Kan, C. Leckie, J. Bailey, J. Markham, R. Chakravorty, Measures for ranking cell trackers without manual validation, Pattern Recognit. 46 (11) (2013) 2849–2859.
[15] X. Yan, I.A. Kakadiaris, S.K. Shah, Modeling local behavior for predicting social interactions towards human tracking, Pattern Recognit. 47 (4) (2014) 1626–1641.
[16] Z. Zhang, O. Faugeras, Three-dimensional motion computation and object segmentation in a long sequence of stereo frames, Int. J. Comput. Vis. 7 (3) (1992) 211–241.

[17] F. Fleuret, J. Berclaz, R. Lengagne, P. Fua, Multi-camera people tracking with a probabilistic occupancy map, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2) (2008) 267–282.
[18] A. Mittal, L. Davis, M 2 tracker: a multi-view approach to segmenting and tracking people in a cluttered scene, Int. J. Comput. Vis. 51 (3) (2003) 189–203.
[19] S. Khan, M. Shah, A multiview approach to tracking people in crowded scenes using a planar homography constraint, in: European Conference on Computer Vision, Graz, Austria: Springer, 2006, pp. 133–146.
[20] S. Butail, N. Manoukis, M. Diallo, J. Ribeiro, T. Lehmann, D. Paley, Reconstructing the flight kinematics of swarming and mating in wild mosquitoes, J. R. Soc. Interface 9 (75) (2012) 2624–2638.
[21] A. Straw, K. Branson, T. Neumann, M. Dickinson, Multi-camera real-time three-dimensional tracking of multiple flying animals, J. R. Soc. Interface 8 (56) (2011) 395–409.
[22] H. Du, D. Zou, Y. Chen, Relative epipolar motion of tracked features for correspondence in binocular stereo, in: IEEE 11th International Conference on Computer Vision, Rio de Janeiro, Brazil: IEEE, 2007, pp. 1–8.
[23] D. Zou, Q. Zhao, H. Wu, Y. Chen, Reconstructing 3d motion trajectories of particle swarms by global correspondence selection, in: IEEE 12th International Conference on Computer Vision, Kyoto, Japan: IEEE, 2009, pp. 1578–1585.
[24] H. Wu, Q. Zhao, D. Zou, Y. Chen, Automated 3d trajectory measuring of large numbers of moving particles, Opt. Express 19 (8) (2011) 7646–7663.
[25] Z. Wu, N. Hristov, T. Hedrick, T. Kunz, M. Betke, Tracking a large number of objects from multiple views, in: IEEE 12th International Conference on Computer Vision, Kyoto, Japan: IEEE, 2009, pp. 1546–1553.
[26] J. Lewis, Fast normalized cross-correlation, in: Vision Interface, vol. 10, 1995, pp. 120–123.
[27] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. 22 (11) (2000) 1330–1334.

Ye Liu received his PhD degree in computer science in 2013 from Fudan University, Shanghai, China and his BS degree from the Department of Computer Science and Technology at Tongji University, Shanghai, China, in 2007. He is now a faculty member with School of Automation, Nanjing University of Posts and Telecommunications. His current research interests include 3D computer vision and pattern recognition, with a special attention to 3D object detection and tracking.

Shuohong Wang received her BS degree from East China University of Science and Technology in 2012. She is now pursuing her PhD degree in School of Computer Science, Fudan University. Her research interests include object detection and multiple object tracking.

Yan Qiu Chen received his PhD degree from Southampton University, United Kingdom in 1995, and his MEng and BEng degrees from Tongji University, Shanghai, China respectively in 1988 and 1985. He is currently a full professor with School of Computer Science of Fudan University, Shanghai, China, and is a member of Fudan University Academic Committee, and a chairman of Computer Science School Academic Committee. He had been a chairman of Department of Communication Science and Engineering from 2004 through 2007, and an associate chairman of Department of Computer Science and Engineering from 2002 through 2004. He was an assistant professor with School of Electrical and Electronic Engineering of Nanyang Technological University, Singapore from 1996 through 2001, and was a postdoctoral research fellow with Glamorgan University, UK in 1995.