A batch asynchronous tracker for wireless smart-camera networks

Sandeep Katragadda, Andrea Cavallaro
Centre for Intelligent Sensing, Queen Mary University of London, UK
Email: {s.katragadda, a.cavallaro}@qmul.ac.uk

Abstract

Distributed tracking in wireless smart-camera networks is affected by varying local processing delays that generally depend on the current scene complexity. As a consequence, each camera makes target information available to the network at different time instants. These unknown delays compound the drifts caused by local clocks and may induce tracking failures when target information is fused. To address this problem, we propose a distributed batch asynchronous tracker for fully connected wireless smart-camera networks. The cameras use the information filter to estimate the target state information and to predict corresponding information of other cameras based on the asynchronous information received from them. Finally, the temporally aligned information is fused. We show that the proposed approach achieves higher tracking accuracy than the state of the art under varying degrees of asynchronism.

1. Introduction

Tracking targets such as people and vehicles using a wireless camera network (WCN) requires cameras to process their captured frames to detect objects of interest. The detections (measurements) may be noisy and the fusion of local detection or tracking results across multiple cameras aims to increase tracking accuracy. Fusion of synchronous information increases tracking accuracy [9, 11]. However, when information is exchanged asynchronously, the accuracy may degrade significantly [10]. Asynchronism is caused by unknown relative *clock offsets* and *processing delays* generated by the local computer-vision pipeline. The former problem is due to the inherent drifts in the local clock frequencies of the nodes [14], which result in cameras capturing frames asynchronously. The latter problem is due to the amount of data to be processed to obtain the target measurements [12, 15, 5]. Moreover, the processing delays may vary from frame to frame and from camera to camera as they depend on the frame size, scene complexity, processing capabilities of the nodes and the object detection algorithms used [13].

In this paper, we propose a tracker that accounts for asynchronism in fully connected WCNs. After every frame capture, each camera performs two phases, namely estimation and fusion. In the *estimation phase*, the camera processes the frame data to obtain the measurement and to estimate the target state using a Bayesian filter. The camera then transmits its local target state information to all the other cameras. If a camera receives a target state from another camera, it stores the received information in a buffer along with the reception time. After a time window around the frame capturing instant, the camera enters the fusion phase. In the *fusion phase*, the camera predicts the target state information of other cameras corresponding to its frame capturing instant using the information received in the time window. The camera fuses the temporally-aligned (i.e corresponding to the same time instant) local and the predicted target information by computing the Kullback-Leibler Average (KLA) [2]. Finally, based on the KLA estimates, the cameras correct their noisy measurements. The software of the proposed method is available at *http://www.eecs.qmul.ac.uk/~andrea/software.htm*.

The paper is organised as follows. Section 2 discusses the related work on distributed asynchronous tracking. Section 3 formulates the asynchronous tracking problem. Section 4 proposes an asynchronous fusion method and the corresponding filter for a linear Gaussian system. The experimental performance evaluation is presented in Section 5. Finally, Section 6 concludes the paper.

2. Related work

This section discusses related work on distributed asynchronous tracking, which is summarised in Table 1.

Distributed asynchronous tracking can be performed using sequential [16] or batch methods [3]. With *sequential methods*, the nodes compute the target state not only when they have their own measurements but also when they receive target information from other nodes. The nodes do not correct their own estimates corresponding to their frame capturing instants using the information received from other nodes. With *batch methods*, the nodes compute the target state only when they have measurements and, during the computation, they consider the information received from other nodes as well. The number of target states computed

Table 1: Related work on distributed asynchronous tracking.

Ref.	Acronym	Characteristics			Fusion method
		considers processing delays	corrects local estimates	one time exchange per measurement	
[16]	SAF			✓	Sequential
[8]	SAF-ED	✓		✓	
[7]	MCAF		✓		Batch (consensus)
[10]	ACAF	✓	✓		
[3]	ABM		✓	✓	Batch
This paper	BAF	✓	✓	✓	

is one per measurement in the batch methods. In contrast, the sequential methods compute every time they receive information from others so they have higher processing load compared to batch methods [6].

The Sequential Asynchronous Filter (SAF) [16] does not consider processing delays and the reception instant of the information is assumed to be the frame capturing instant of the sender. The SAF with estimated delays (SAF-ED) [8] extends SAF to model and estimate the delays. When a node receives state information from another node, it estimates the delay and thereby the frame capturing instant corresponding to the received information. The node predicts the local target information corresponding to the capturing instant of the sender based on the previously known local information. Finally, the temporally aligned information is fused.

The Maximum Consensus-based Asynchronous Filter (MCAF) updates the local estimates with the most certain state among the local and received estimates [7]. As there is no fusion, MCAF does not reduce the uncertainty of the estimates. The Average Consensus-based Asynchronous Filter (ACAF) [10] uses predictions to temporally align the information before fusion. With ACAF, nodes do not consider the target information (of other cameras) received before capturing their frames. ACAF requires multiple iterations of information exchange between captures to guarantee the reception after capturing. As consensus methods process the information of all cameras together, we classify them as batch methods. In the Asynchronous Batch Method (ABM) [3], each node assumes that processing and transmission delays are negligible. However, due to the significant delays in camera networks, the capturing instant of the sender is different from the reception instant. This paper proposes a batch method, *the Batch Asynchronous Filter* (BAF), that considers processing delays during fusion.

3. Problem formulation

Let $\mathbf{C} = \left\{C^1, C^2, ..., C^N\right\}$ be a fully connected WCN with N calibrated cameras with local computation capabilities that track a target on a common ground plane. We assume that cameras communicate without packet losses and communication delays. Let k be the index for the local time of C^i when capturing a frame. C^i performs target detection to obtain on the image plane measurement $\mathbf{z}_k^i$ and τ_k^i is the corresponding processing delay. We assume that the measurements are noisy but not false (i.e. no false positive or false negative detections).

Let the target state be $\mathbf{x}_k^i = \left[x_k^i\ y_k^i\ \dot{x}_k^i\ \dot{y}_k^i\right]^\top$, where $[x_k^i\ y_k^i]^\top$ and $[\dot{x}_k^i\ \dot{y}_k^i]^\top$ are the position and velocity of the target on the ground plane respectively, when the local time of the camera C^i is k. Let α^{ij} be the relative offset in the capturing instants of C^i and C^j. Our objective is to correct the noisy measurements $\mathbf{z}_k^{1:N}$ by fusing the local target state information of all the cameras in the presence of asynchronous captures and varying processing delays.

4. Proposed approach

This section proposes a batch method to fuse asynchronous information. The method can be applied to linear or non-linear Gaussian systems by using an appropriate filter. It is followed by a filter implementation for linear and Gaussian systems using the Information Filter [4].

4.1. Batch asynchronous fusion

In the *estimation phase*, each camera C^i processes the frame captured at k to obtain the measurement $\mathbf{z}_k^i$ and a local Bayesian filter to compute the target probability density function (*pdf*) $p(\mathbf{x}_k^{i,i}|\mathbf{z}_{1:k}^i)$ using the known *pdf* $p(\hat{\mathbf{x}}_{k'}^i|\mathbf{z}_{1:k'}^{1:N})$ corresponding to its previous frame capturing instant k', the state transition *pdf* $p(\mathbf{x}_k^{i,i}|\hat{\mathbf{x}}_{k'}^i)$ and the measurement likelihood $p(\mathbf{z}_k^i|\mathbf{x}_k^{i,i})$. The subscript k indicates the local time of the camera whose index is the first superscript. The second superscript indicates the index of the camera the information corresponds to. Each camera C^i completes its estimation phase at $k + \tau_k^i$. If the cameras capture synchronously, the local *pdf*s $p(\mathbf{x}_k^i|\mathbf{z}_{1:k}^i), \forall C^i \in \mathbf{C}$ correspond to the same time instant (at k). As the network is fully connected, each camera C^i receives all the *pdf*s $p(\mathbf{x}_k^j|\mathbf{z}_{1:k}^j), \forall C^j \in \mathbf{C}$. In the *fusion phase*, C^i fuses them by computing their KLA [2]

$$p(\hat{\mathbf{x}}_k^i|\mathbf{z}_{1:k}^{1:N}) = \frac{\prod_{j=1}^{N} p(\mathbf{x}_k^{j,j}|\mathbf{z}_{1:k}^j)^{\frac{1}{N}}}{\int \prod_{j=1}^{N} p(\mathbf{x}_k^{j,j}|\mathbf{z}_{1:k}^j)^{\frac{1}{N}} d\mathbf{x}}, \forall C^i \in \mathbf{C}, \qquad (1)$$

which is the same at all cameras.

In reality, the cameras capture asynchronously due to the varying frame processing delays so the *pdf*s received by each camera C^i might not correspond to its capturing instant k. The proposed approach temporally aligns the asynchronous local *pdf*s before computing their KLA. Each camera C^i predicts the *pdf* of other cameras $p(\tilde{\mathbf{x}}_k^{i,j}|\mathbf{z}_{1:k}^j)$, $\forall C^j \in \mathbf{C} \backslash \left\{C^i\right\}$ corresponding to its frame capturing instant k. There are two possible *communication modes* to achieve this. The modes differ in the type of information

exchanged in the network. In the first mode, the cameras exchange the predicted target *pdf*s corresponding to their transmission instants $k+\tau_k^i$. In the second mode, the cameras exchange the estimated target *pdf*s corresponding to their frame capturing instants k and also the local-delays τ_k^i. The receiving cameras know the capturing instant of the sender by subtracting the received delay from the reception instant and the knowledge makes the cameras predict more certain target *pdf*s of other cameras compared to the first mode. Due to the additional transmission of local-delay knowledge, the communication load is one scalar higher compared to the previous mode. We therefore name the two modes as *lower load* mode and *higher load* mode.

In the lower load mode, the predicted *pdf* to transmit $p(\tilde{\mathbf{x}}^{i,i}_{k+\tau_k^i}|\mathbf{z}^i_{1:k})$ is computed as

$$p(\tilde{\mathbf{x}}^{i,i}_{k+\tau_k^i}|\mathbf{z}^i_{1:k}) = \int p(\mathbf{x}^{i,i}_{k+\tau_k^i}|\mathbf{x}^{i,i}_k)p(\mathbf{x}^{i,i}_k|\mathbf{z}^i_{1:k})d\mathbf{x}^{i,i}_k. \quad (2)$$

The predicted *pdf* represents the knowledge of C^i about the target state corresponding to the transmission instant $k+\tau^i$. C^i broadcasts the predicted *pdf* $p(\tilde{\mathbf{x}}^{i,i}_{k+\tau_k^i}|\mathbf{z}^i_{1:k})$ to all other cameras. C^i receives similar *pdf*s from other cameras. Each camera C^i fuses the *pdf*s received in a time window $\mathbf{K}^i_k = [k-\alpha_1, k+\alpha_2]$, ($\alpha_1 > 0$ and $\alpha_2 > \tau^i_k$) around its capturing instant k. C^i enters the *fusion phase* at the end of the window ($k+\alpha_2$). Let $k'' \in \mathbf{K}^i_k$ be a reception instant of the *pdf* $p(\tilde{\mathbf{x}}^{j,j}_{k+\tau_k^j}|\mathbf{z}^j_{1:k})$ from C^j. C^i considers the *pdf* as $p(\tilde{\mathbf{x}}^{i,j}_{k''}|\mathbf{z}^j_{1:k})$. C^i predicts the *pdf* of C^j corresponding to its capturing instant k based on the received *pdf* $p(\tilde{\mathbf{x}}^{i,j}_{k''}|\mathbf{z}^j_{1:k})$ as

$$p(\tilde{\mathbf{x}}^{i,j}_{k}|\mathbf{z}^j_{1:k}) = \int p(\mathbf{x}^{i,j}_{k}|\mathbf{x}^{i,j}_{k''})p(\tilde{\mathbf{x}}^{i,j}_{k''}|\mathbf{z}^j_{1:k})d\tilde{\mathbf{x}}^{i,j}_{k''}. \quad (3)$$

In the higher load mode, C^i broadcasts at $k+\tau^i_k$ its local *pdf* $p(\mathbf{x}^{i,i}_k|\mathbf{z}^i_{1:k})$ and the delay τ^i_k to all other cameras. C^i receives similar information from other cameras. C^i considers the received *pdf* from C^j at $k'' \in \mathbf{K}^i_k$ as $p(\mathbf{x}^{i,j}_{k''-\tau^j_k}|\mathbf{z}^j_{1:k})$. C^i predicts the *pdf* of C^j corresponding to its capturing instant k based on the received *pdf* $p(\mathbf{x}^{i,j}_{k''-\tau^j_k}|\mathbf{z}^j_{1:k})$ as

$$p(\tilde{\mathbf{x}}^{i,j}_{k}|\mathbf{z}^j_{1:k}) = \int p(\mathbf{x}^{i,j}_{k}|\mathbf{x}^{i,j}_{k''-\tau^j_k})p(\mathbf{x}^{i,j}_{k''-\tau^j_k}|\mathbf{z}^j_{1:k})d\mathbf{x}^{i,j}_{k''-\tau^j_k}. \quad (4)$$

In both modes, C^i considers the predicted *pdf* $p(\tilde{\mathbf{x}}^{i,j}_k|\mathbf{z}^j_{1:k})$ as the knowledge of C^j corresponding to the capturing instant k of C^i. As the local *pdf* $p(\mathbf{x}^{i,i}_k|\mathbf{z}^i_{1:k})$ and the predicted *pdf*s $p(\tilde{\mathbf{x}}^{i,j}_k|\mathbf{z}^j_{1:k}), \forall C^j \in \mathbf{C}\backslash\{C^i\}$, correspond to the same time instant (capturing instant k of C^i), they are now temporally aligned. C^i now fuses the temporally aligned *pdf*s by computing their KLA [2] as

$$p(\hat{\mathbf{x}}^i_k|\mathbf{z}^{1:N}_{1:k}) = \frac{p(\mathbf{x}^{i,i}_k|\mathbf{z}^i_{1:k})^{\frac{1}{N}} \prod_{j=1,j\neq i}^{N} p(\tilde{\mathbf{x}}^{i,j}_k|\mathbf{z}^j_{1:k})^{\frac{1}{N}}}{\int p(\mathbf{x}^{i,i}_k|\mathbf{z}^i_{1:k})^{\frac{1}{N}} \prod_{j=1,j\neq i}^{N} p(\tilde{\mathbf{x}}^{i,j}_k|\mathbf{z}^j_{1:k})^{\frac{1}{N}} d\mathbf{x}}. \quad (5)$$

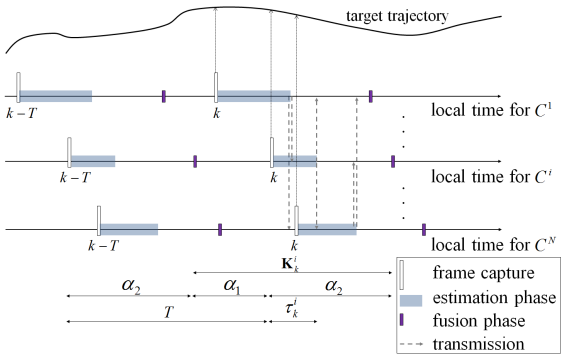


Figure 1: The timeline of the batch asynchronous fusion. KEY – C^i: camera i, τ^i_k: delay of C^i to process its frame captured at k, $T = \alpha_1 + \alpha_2$: inter-frame interval, $\mathbf{K}^i_k = [k-\alpha_1, k+\alpha_2]$: reception time window corresponding to the frame capturing instant k of C^i.

To avoid fusing information from subsequent estimation phases, subsequent captures must be well separated. To achieve this, the cameras capture their next frames after $T = \alpha_1 + \alpha_2$. The higher the values of α_1 and α_2, the higher the inter-frame interval (fewer captures), and the lower the values of α_1 and α_2, the smaller the window $\mathbf{K}^i_k$ and the information of some cameras is not considered for fusion.

In the case of partial asynchronism, where there is an upper limit on the relative offsets [7, 10], $\alpha_1 = \alpha^{max} - \tau^{min}$ and $\alpha_2 = \alpha^{max} + \tau^{max}$ are the earliest and the latest possible reception instants. Here, $\alpha^{max} = \max_{i,j}\{\alpha^{ij}\}$ is the maximum relative offset and $\tau^{max} = \max_{i,k}\{\tau^i_k\}$ and $\tau^{min} = \min_{i,k}\{\tau^i_k\}$ are the maximum and minimum processing delays, respectively. Figure 1 shows the timeline of the proposed approach.

Finally, C^i computes the updated measurement $\hat{\mathbf{z}}^i_k$ of the target using the KLA estimate $p(\hat{\mathbf{x}}^i_k|\mathbf{z}^{1:N}_{1:k})$ and the known camera measurement model.

4.2. Linear and Gaussian case

Based on the Batch asynchronous fusion described above, we now derive the *Batch Asynchronous Filter* (BAF) for linear and Gaussian systems using the Information Filter (IF) [4] as the local Bayesian filter. Let the target follow a linear motion model described as

$$\mathbf{x}^i_{k+\Delta k} = \mathbf{F}(k, k+\Delta k)\mathbf{x}^i_k + \mathbf{w}(k, k+\Delta k), \quad (6)$$

where $\mathbf{F}(k, k+\Delta k)$ is the matrix describing the state transition from k to $k+\Delta k$ and $\mathbf{w}(k, k+\Delta k)$ is the cumulative effect of the process noise from k to $k+\Delta k$, which is assumed to be Gaussian noise with zero mean and covariance matrix $\mathbf{Q}(k, k+\Delta k)$. Let the measurement model of each camera C^i be defined as $\mathbf{z}^i_k = \mathbf{H}^i\mathbf{x}^i_k + \mathbf{v}^i_k$, where $\mathbf{v}^i_k$ is the measurement noise vector, which is assumed to be Gaussian with

zero mean and covariance matrix $\mathbf{R}^i, \forall C^i$. In the estimation phase of BAF, each camera C^i computes the target *pdf* using the IF. The *pdf* is represented by the information pair $\left(\mathbf{y}_k^{i,i}, \mathbf{Y}_k^{i,i}\right)$ containing the information vector $\mathbf{y}_k^{i,i}$ and the information matrix $\mathbf{Y}_k^{i,i}$. The IF uses the previously known information pair $\left(\hat{\mathbf{y}}_{k-T}^i, \hat{\mathbf{Y}}_{k-T}^i\right)$ and the current measurement $\mathbf{z}_k^i$.

In the lower load mode, C^i predicts the target information corresponding to the transmission instant $k+\tau_k^i$ as

$$\begin{aligned}\tilde{\mathbf{Y}}_{k+\tau_k^i}^{i,i} &= (\mathbf{F}(k, k+\tau_k^i)\mathbf{Y}_k^{i,i^{-1}}\mathbf{F}(k, k+\tau_k^i)^\top + \mathbf{Q}(k, k+\tau_k^i))^{-1},\\ \tilde{\mathbf{y}}_{k+\tau_k^i}^{i,i} &= \tilde{\mathbf{Y}}_{k+\tau_k^i}^{i,i}\mathbf{F}(k, k+\tau_k^i)\left(\mathbf{Y}_k^{i,i^{-1}}\mathbf{y}_k^{i,i}\right),\end{aligned} \tag{7}$$

and transmits the predicted information $\left(\tilde{\mathbf{y}}_{k+\tau_k^i}^{i,i}, \tilde{\mathbf{Y}}_{k+\tau_k^i}^{i,i}\right)$ to all cameras at $k+\tau_k^i$. C^i receives the information pair $\left(\tilde{\mathbf{y}}_{k+\tau_k^j}^{j,j}, \tilde{\mathbf{Y}}_{k+\tau_k^j}^{j,j}\right)$ from C^j at $k'' \in \mathbf{K}_k^i$. C^i considers the information as $\left(\tilde{\mathbf{y}}_{k''}^{i,j}, \tilde{\mathbf{Y}}_{k''}^{i,j}\right)$. In the fusion phase (at $k+\alpha_2$), C^i predicts the information of $C^j, \forall C^j \in \mathbf{C}$, corresponding to its frame capturing instant k as

$$\begin{aligned}\tilde{\mathbf{Y}}_k^{i,j} &= \left(\mathbf{F}(k'', k)\tilde{\mathbf{Y}}_{k''}^{i,j^{-1}}\mathbf{F}(k'', k)^\top + \mathbf{Q}(k'', k)\right)^{-1},\\ \tilde{\mathbf{y}}_k^{i,j} &= \tilde{\mathbf{Y}}_k^{i,j}\mathbf{F}(k'', k)\left(\tilde{\mathbf{Y}}_{k''}^{i,j^{-1}}\tilde{\mathbf{y}}_{k''}^{i,j}\right).\end{aligned} \tag{8}$$

In the higher load mode, C^i transmits its local information $\left(\mathbf{y}_k^{i,i}, \mathbf{Y}_k^{i,i}\right)$ and its local processing delay τ_k^i to all cameras at $k+\tau_k^i$ without any prediction so that the other cameras can know the actual estimate and the corresponding capturing instant of C^i. Let $k'' \in \mathbf{K}_k^i$ be the time instant when C^i received the information pair $\left(\mathbf{y}_k^{j,j}, \mathbf{Y}_k^{j,j}\right)$ and τ_k^j from C^j. As we assume no communication delay, C^i considers the information as $\left(\mathbf{y}_{k''-\tau_k^j}^{i,j}, \mathbf{Y}_{k''-\tau_k^j}^{i,j}\right)$. In the fusion phase (at $k+\alpha_2$), C^i predicts the information of C^j, $\forall C^j \in \mathbf{C}$, corresponding to its frame capturing instant k as

$$\begin{aligned}\tilde{\mathbf{Y}}_k^{i,j} &= (\mathbf{F}(k''-\tau_k^j, k)\mathbf{Y}_{k''-\tau_k^j}^{i,j^{-1}}\mathbf{F}(k''-\tau_k^j, k)^\top + \mathbf{Q}(k''-\tau_k^j, k))^{-1},\\ \tilde{\mathbf{y}}_k^{i,j} &= \tilde{\mathbf{Y}}_k^{i,j}\mathbf{F}(k''-\tau_k^j, k)\left(\mathbf{Y}_{k''-\tau_k^j}^{i,j^{-1}}\mathbf{y}_{k''-\tau_k^j}^{i,j}\right).\end{aligned} \tag{9}$$

In both modes, the KLA estimate of the Gaussian *pdf*s can indeed be obtained by the average of the information terms [2] so C^i computes the fused estimate as

$$\begin{aligned}\hat{\mathbf{Y}}_k^i &= \frac{1}{N}\left(\mathbf{Y}_k^{i,i} + \sum_{j=1, j\neq i}^{N}\tilde{\mathbf{Y}}_k^{i,j}\right),\\ \hat{\mathbf{y}}_k^i &= \frac{1}{N}\left(\mathbf{y}_k^{i,i} + \sum_{j=1, j\neq i}^{N}\tilde{\mathbf{y}}_k^{i,j}\right).\end{aligned} \tag{10}$$

Algorithm 1 The Batch Asynchronous Filter (BAF) running in camera C^i

Notations:
$\mathbf{z}_k^i, \hat{\mathbf{z}}_k^i$: the collected and the corrected measurements of C^i corresponding to its local time k
$\left(\mathbf{y}_k^{i,j}, \mathbf{Y}_k^{i,j}\right)$: local information pair of C^j corresponding to the local time k of C^i
$\left(\tilde{\mathbf{y}}_k^{i,j}, \tilde{\mathbf{Y}}_k^{i,j}\right)$: predicted information pair of C^j corresponding to the local time k of C^i
$\left(\hat{\mathbf{y}}_k^i, \hat{\mathbf{Y}}_k^i\right)$: corrected information pair of C^i corresponding to its local time k (i.e. after fusion)
τ_k^i : processing delay of C^i corresponding to its frame captured at local time k
$\mathbf{K}_k^i$: reception time window of C^i at k
T : inter-frame interval
$\mathbf{H}^i$: State-to-measurement transition matrix of C^i

Input: $\hat{\mathbf{y}}_{k-T}^i, \hat{\mathbf{Y}}_{k-T}^i, \mathbf{z}_k^i$

Compute local information pair $\left(\mathbf{y}_k^{i,i}, \mathbf{Y}_k^{i,i}\right)$ using the information filter
Switch(Communication mode **)**
Case lower load **:**
  Predict the information pair $\left(\tilde{\mathbf{y}}_{k+\tau_k^i}^{i,i}, \tilde{\mathbf{Y}}_{k+\tau_k^i}^{i,i}\right)$ Use (7)
  Send $\left(\tilde{\mathbf{y}}_{k+\tau_k^i}^{i,i}, \tilde{\mathbf{Y}}_{k+\tau_k^i}^{i,i}\right)$ to $\forall C^j \in \mathbf{C}\backslash\left\{C^i\right\}$
  Receive $\left(\tilde{\mathbf{y}}_{k''}^{i,j}, \tilde{\mathbf{Y}}_{k''}^{i,j}\right)$ from $\forall C^j \in \mathbf{C}\backslash\left\{C^i\right\}, k'' \in \mathbf{K}_k^i$
  Predict information pair of C^j $\left(\tilde{\mathbf{y}}_k^{i,j}, \tilde{\mathbf{Y}}_k^{i,j}\right)$ Use (8)
Case higher load **:**
  Send $\left(\mathbf{y}_k^{i,i}, \mathbf{Y}_k^{i,i}, \tau_k^i\right)$ to $\forall C^j \in \mathbf{C}\backslash\left\{C^i\right\}$
  Receive $\left(\mathbf{y}_{k''-\tau_k^j}^{i,j}, \mathbf{Y}_{k''-\tau_k^j}^{i,j}, \tau_k^j\right)$ from $\forall C^j \in \mathbf{C}\backslash\left\{C^i\right\}, k'' \in \mathbf{K}_k^i$
  Predict information pair of C^j $\left(\tilde{\mathbf{y}}_k^{i,j}, \tilde{\mathbf{Y}}_k^{i,j}\right)$ Use (9)
end Switch
Compute $(\hat{\mathbf{y}}_k^i, \hat{\mathbf{Y}}_k^i)$ by information fusion Use (10)
Compute $\hat{\mathbf{z}}_k^i = \mathbf{H}^i\hat{\mathbf{Y}}_k^{i^{-1}}\hat{\mathbf{y}}_k^i$
Output: $\hat{\mathbf{y}}_k^i, \hat{\mathbf{Y}}_k^i$ and $\hat{\mathbf{z}}_k^i$

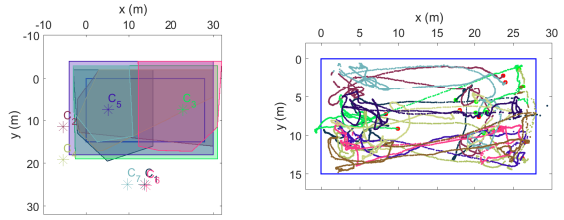
The local measurement $\mathbf{z}_k^i$ is finally corrected as $\hat{\mathbf{z}}_k^i = \mathbf{H}^i\hat{\mathbf{Y}}_k^{i^{-1}}\hat{\mathbf{y}}_k^i$. Algorithm 1 summarises the proposed Batch Asynchronous Filter.

5. Results

5.1. Evaluation setup

We compare the proposed Batch Asynchronous Filter that operates using lower communication load (BAF-1) and using higher communication load (BAF-2) with five filters, namely the Asynchronous Batch Method (ABM) [3], the Sequential Asynchronous Filter (SAF) [16], the Sequential Asynchronous Filter with estimated delays (SAF-ED) [8], the Maximum Consensus-based Asynchronous Filter (MCAF) [7], the Average Consensus-based Asynchronous Filter (ACAF) [10] and the distributed filter without fusion (No fusion). As MCAF and ACAF work only with partial asynchronism, we also consider this case for a fair comparison. We use the ground truth processing delay knowledge for the estimated delays in SAF-ED.

We use the APIDIS dataset [1], which is captured with $N = 7$ cameras whose positions and field of views (FoVs) are known (Figure 2a). The camera network monitors a $30m \times 20m$ basketball court. We consider the trajectories of $P = 10$ players whose ground-truth positions on the ground plan are known for a duration of 1500 time steps



(a) Field of views (b) 10 ground plane trajectories

Figure 2: Illustration of the APIDIS dataset [1] setup in terms of camera viewpoints and trajectories of players.

(Figure 2b). Each time step corresponds to $40ms$. We simulate the asynchronous captures and processing delays by skipping some frames. We define the state transition matrix $\mathbf{F}(\cdot)$ and the process noise covariance $\mathbf{Q}(\cdot)$ as [8]

$$\begin{aligned}\mathbf{F}(k,k+\Delta k) &= \begin{bmatrix} \mathbf{I}_2 & (\Delta k)\mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix}, \\ \mathbf{Q}(k,k+\Delta k) &= q^2 \begin{bmatrix} d(\Delta k,3)\mathbf{I}_2 & d(\Delta k,2)\mathbf{I}_2 \\ d(\Delta k,2)\mathbf{I}_2 & d(\Delta k,1)\mathbf{I}_2 \end{bmatrix},\end{aligned} \tag{11}$$

where $\mathbf{I}_2$ and $\mathbf{0}_2$ are the 2×2 identity and zero matrices respectively, $d(\Delta k, n) = \frac{|\Delta k|^n}{n}$ and $q = 10$ is the noise intensity in the unit interval. $\mathbf{H}^i = [\mathbf{I}_2\ \mathbf{0}_2]$ and $\mathbf{R}^i = 60\mathbf{I}_2, \forall C^i$. The random processing delays are $\tau_k^i \in \mathcal{U}\{0, \tau^{max}\}$, $\forall C^i \in \mathbf{C}$. Similarly, the random relative offsets among cameras are: $\alpha^{ij} \in \mathcal{U}\{0, \alpha^{max}\}$, $\forall C^i, C^j \in \mathbf{C}, i \neq j$. We use a single iteration of information exchange per measurement. We track each player using $M = 20$ Monte Carlo simulation runs. Each run uses a different set of processing delays, relative offsets and measurements.

Let $\mathcal{T}^{i,r} \subset [1, 1500]$ be the set of frame capturing instants of C^i during the r^{th} run and $\mathcal{P}_k^i \subset [1, P]$ be the set of players in the FoV of C^i at k. We define as performance measure the *average root mean square error* (ϵ) of P players' locations on the image planes of the cameras:

$$\epsilon = \sqrt{\frac{1}{MN}\sum_{r=1}^{M}\sum_{i=1}^{N}\frac{1}{|\mathcal{T}^{i,r}|}\sum_{k\in\mathcal{T}^{i,r}}\frac{1}{|\mathcal{P}_k^i|}\sum_{p\in\mathcal{P}_k^i}||\hat{\mathbf{z}}_k^i(r,p) - \zeta_k^i||_2^2}. \tag{12}$$

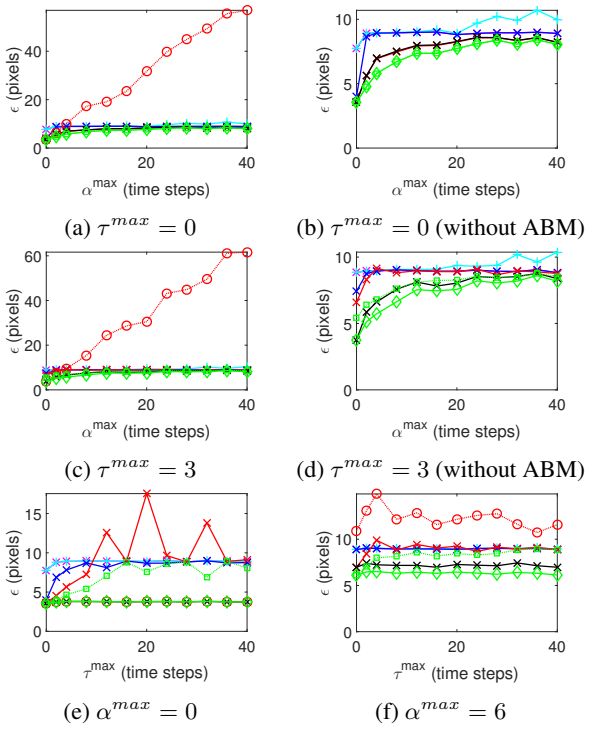
Here, $\hat{\mathbf{z}}_k^i(r,p)$ is the estimated location of player p in the image plane of camera C^i at k during the r^{th} run and ζ_k^i is the corresponding image plane ground truth. We analyse the error ϵ with increasing level of relative offset α^{max} for two fixed maximum processing delays $\tau^{max} = 0$ (no delay) and $\tau^{max} = 3$. We also analyse ϵ with increasing level of processing delay τ^{max} for two fixed maximum relative offsets $\alpha^{max} = 0$ (synchronous case) and $\alpha^{max} = 6$ (asynchronous case).

5.2. Discussion

With synchronism ($\alpha^{max} = 0$) and without processing delays ($\tau^{max} = 0$), we can identify two groups with similar performance: The group BAF-1, BAF-2, ACAF, SAF and SAF-ED has higher accuracy than the group MCAF and No fusion (Figures 3b and 3e).

With asynchronism ($\alpha^{max} > 0$), BAF-2 outperforms all other methods irrespective of the delays (Figures 3b, 3d and 3f). Unlike BAF-1 and BAF-2, the sequential methods (SAF and SAF-ED) do not correct the estimates corresponding to the frame capturing instants using the delayed information. BAF-1 and BAF-2 outperform ACAF and MCAF, because in ACAF the nodes do not consider the information received before their frame capturing instants and in MCAF nodes do not fuse the information but select the most certain information that might correspond to a different time step.

With synchronism ($\alpha^{max} = 0$) and with processing



(a) $\tau^{max} = 0$ (b) $\tau^{max} = 0$ (without ABM)

(c) $\tau^{max} = 3$ (d) $\tau^{max} = 3$ (without ABM)

(e) $\alpha^{max} = 0$ (f) $\alpha^{max} = 6$

Figure 3: Average Root Mean Square tracking Error (ϵ) with varying maximum relative offset α^{max} and maximum processing delay τ^{max}. One time step $\approx 40ms$. The algorithms under analysis are the proposed Batch Asynchronous Filters (- - - -BAF-1 and ——BAF-2), the Asynchronous Batch Method (- - - -ABM) [3], the Sequential Asynchronous Filter (——SAF) [16], the Sequential Asynchronous Filter with estimated delays (——SAF-ED) [8], the Maximum Consensus-based Asynchronous Filter (——MCAF) [7], the Average Consensus-based Asynchronous Filter (——ACAF) [10] and the distributed filter that never fuses (——No fusion).

delays ($\tau^{max} > 0$), BAF-1 and BAF-2 outperform all other methods except SAF-ED (Figure 3e). This is because the accuracy of the KLA estimate decreases if the uncertainty of the *pdf*s increases. In BAF-1, the information of other cameras undergoes two predictions thereby reducing the uncertainty: the first by the sender C^j from its frame capturing instant k to its transmission instant $k + \tau_k^j$ and second by the receiver C^i from the reception instant $k + \tau_k^j$ to its frame capturing instant k. In SAF-ED and BAF-2, based on the known delay, the receiving node computes the capturing instant of the sender. As all the capturing instants are the same, the prediction does not affect the fusion. However, the performance of SAF-ED depends on the correctness of the estimated delays. Moreover, if a node has no target observations, with ACAF and MCAF the node sends predicted target information, whereas in SAF, SAF-ED, BAF-1 and BAF-2, the node does not send any information so the tracking accuracy of SAF, SAF-ED and BAF-1 and BAF-2 is not affected. In the case of delays ($\tau^{max} > 0$), it is better to skip fusion instead of using MCAF (Figure 3b, 3d, 3e and 3f). Note that the errors are upper-bounded by the error with no fusion. If the captures are synchronous ($\alpha^{max} = 0$), ABM and BAF-2 outperform all others (Figure 3e). If the captures are asynchronous ($\alpha^{max} > 0$), ABM has the highest error among all methods, irrespective of the delays (Figures 3a, 3c, 3f).

BAF-2 is the best choice when the network is fully connected. When the network is with limited connectivity, ACAF is the best choice but ACAF discards the information received prior to the frame capturing instant. To handle this problem, in ACAF each node sends its information multiple times between subsequent captures such that the neighbours may receive the information after their frame capturing instants. This increases the communication cost.

6. Conclusions

We proposed a batch method for asynchronous tracking in fully connected wireless camera networks where each camera predicts the target information of other cameras to time-align it with its local information. The cameras consider and process only the information received in time windows around their local frame capturing instants. Importantly the proposed method considers the information received both before and after its frame capturing instants. Moreover, the proposed method corrects the local estimates with respect to their frame capturing instants by fusing the time-aligned information. The proposed tracking approach outperforms state-of-the-art methods in terms of image-plane error.

As future work, we will model the delays to aid predictions and will validate the proposed approach with a physical smart-camera network.

References

[1] APIDIS dataset. `http://sites.uclouvain.be/ispgroup/index.php/Softwares/APIDIS`. Last accessed: 2017-06-26.

[2] G. Battistelli and L. Chisci. Kullback-Leibler average, consensus on probability densities, and distributed state estimation with guaranteed stability. *Automatica*, 50(3):707–718, Mar 2014.

[3] J. Beaudeau, M. F. Bugallo, and P. M. Djurić. Target tracking with asynchronous measurements by a network of distributed mobile agents. In *Proc. of ICASSP*, pages 3857–3860, Mar 2012.

[4] D. Casbeer and R. Beard. Distributed information filtering using consensus filters. In *Proc. of the American Control Conf.*, pages 1882–1887, 2009.

[5] G. Chen, P. L. St-Charles, W. Bouachir, G. A. Bilodeau, and R. Bergevin. Reproducible evaluation of pan-tilt-zoom tracking. In *Proc. of ICIP*, pages 2055–2059, Sep 2015.

[6] A. F. García-Fernández and J. Grajal. Asynchronous particle filter for tracking using non-synchronous sensor networks. *Signal Processing*, 91(10):2304–2313, Oct 2011.

[7] S. Giannini, A. Petitti, D. D. Paola, and A. Rizzo. Asynchronous consensus-based distributed target tracking. In *Proc. of the IEEE Conf. on Decision and Control*, pages 2006–2011, Dec 2013.

[8] O. Hlinka, F. Hlawatsch, and P. M. Djurić. Distributed sequential estimation in asynchronous wireless sensor networks. *IEEE Signal Processing Let.*, 22(11):1965–1969, Nov 2015.

[9] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information weighted consensus filters and their application in distributed camera networks. *IEEE Trans. on Automatic Control*, 58(12):3112–3125, Dec 2013.

[10] S. Katragadda, C. S. Regazzoni, and A. Cavallaro. Average consensus-based asynchronous tracking. In *Proc. of ICASSP*, Mar 2017.

[11] S. Katragadda, J. C. SanMiguel, and A. Cavallaro. The costs of fusion in smart camera networks. In *Proc. of the Int. Conf. on Distributed Smart Cameras*, Nov 2014.

[12] S. Persa and P. P. Jonker. Real-time image processing architecture for robot vision. *Proc. SPIE, Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision*, 4197:221–228, 2000.

[13] J. C. SanMiguel and A. Cavallaro. Energy consumption models for smart-camera networks. *IEEE Trans. on Circuits and Systems for Video Technology*, 2016, DOI: 10.1109/TCSVT.2016.2593598.

[14] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45–50, Jul 2004.

[15] S. F. Stefánsson, B. Jónsson, and K. R. Thórisson. A YARP-based architectural framework for robotic vision applications. In *Proc. of the fourth Int. Conf. on Computer Vision Theory and Applications*, pages 65–68, Feb 2009.

[16] G. Zhu, F. Zhou, L. Xie, R. Jiang, and Y. Chen. Sequential asynchronous filters for target tracking in wireless sensor networks. *IEEE Sensors Jour.*, 14(9):3174–3182, Sep 2014.