Distributed and decentralized multi-camera tracking

Murtaza Taj and Andrea Cavallaro

In recent years the decreasing cost of cameras and advances in miniaturization have favoured the deployment of large-scale camera networks. This growing number of cameras enables new signal processing applications that use cooperatively multiple sensors over wide areas. In particular, object tracking is an important step in many applications related to security, traffic monitoring and event recognition. Such applications require the optimal trade-off between accuracy, communication and computing across the network. The costs associated to communication and computing depend on the type and amount of cooperation performed among cameras for information gathering, sharing and processing to validate decisions as well as to rectify (or to reduce) estimation errors and uncertainties. In this survey we discuss data fusion and tracking methods for camera networks and compare their performance. In particular we cover decentralized and distributed trackers and the challenges to be addressed for the design of accurate and energy-efficient algorithms.

I. Introduction

Let us consider a network $C = \{C_1, \cdots, C_c, \cdots, C_N\}$ of N cameras monitoring T targets. Let the state[1] of target i at time k be defined as $\mathbf{x}_k^{\nu,i}$, where $\nu \in \{c, \pi\}$ represents either the c^{th} camera view or an hypothetical top view π that for simplicity we will consider to be a plane. The goal is to estimate the target state $\mathbf{x}_k^{\nu,i}$ by fusing the data gathered from the cameras. Target state estimation on ν aims to associate noisy measurements $Z_k^{\nu,i} = \{z_1^{\nu,i}, \cdots, z_k^{\nu,i}\}$ belonging to the same object over time to obtain the trajectory $X_k^{\nu,i} = \{\mathbf{x}_1^{\nu,i}, \cdots, \mathbf{x}_k^{\nu,i}\}$ for each object i.

The amount and type of information sharing for state estimation differs from tracker to tracker. Depending on the algorithm, type of cameras in the network and on the strategy adopted for data fusion, multi-camera trackers incur different computation and communication costs. Moreover, the data gathering (fusion) strategy has a significant influence on the scalability of the network and on the communication cost, thus affecting the applicability of a tracker. Multi-camera trackers can be categorized, based on inter-sensor communication [1], in three main groups, namely centralized, decentralized and distributed tracking.

Centralized tracking is performed in a single node that receives (raw or processed) data from each camera in the network. Although centralized approaches can exploit directly single-camera trackers on the fused data [2], the presence of a single global fusion center [3]–[5] leads to high data-transfer rates and to a lack of scalability and energy efficiency.

In *decentralized tracking* cameras are grouped into clusters and member nodes communicate with their local fusion centers only [6]–[8]. The communication overhead is reduced by limiting the cooperation within each cluster and among fusion centers. Object features are extracted in each camera view and then projected to the fusion center for multi-camera tracking. Finally, fusion centers communicate with each other to hand-off tracking estimates over the network [9]. Energy efficiency can be improved by selectively activating only cluster members that are expected to observe the targets of interest [10].

To further increase scalability and to reduce communication costs, *distributed tracking* operates without local fusion centers. The estimates generated in a camera are transmitted to its immediate neighbours only. The received estimates are used to refine the next-camera estimates and these refined estimates are then transmitted to the next neighbour [10]–[16]. This process is completed after a pre-defined number of steps, after all cameras viewing the target are visited or when the uncertainty has decreased below a desired value.

In this survey we discuss decentralized[2] and distributed multi-camera tracking approaches and, for simplicity and clarity, we restrict ourselves to stationary cameras and targets moving on planar surfaces. Surveys on active camera networks and on other general issues related to multi-view multi-target tracking can be found in [17]–[21].

Before comparing specific tracking approaches, we will cover common algorithmic steps that are essential stages prior to applying a multi-camera tracking algorithm, such as calibration, synchronization and the selection of fusion centers.

II. Calibration and synchronization

To estimate the trajectory of objects moving across the network, cameras share various types of data such as target measurements (position, velocity, size, contour and appearance), target states, estimate uncertainties (covariance matrices) and other derived measures such as amount of activity in a camera or visibility of certain features. To efficiently exploit this shared information, cameras have to be aware of each other and to recognize where and at what abstraction level the information is fused.

[1]The elements composing the state depend on the application. In the simplest case, the state is defined by the position of the target. In more complex cases, the state contains other elements such as for example shape and velocity parameters.

[2]Centralized tracking is considered a special case of decentralized tracking with a single fusion center.

Multi-camera fusion can be performed through correspondence between measurements $z_k^{c,i}$ (e.g. positions, bounding boxes, blobs) or trajectories. In decentralized camera networks, *measurement correspondence* maps the features (measurements) $Z_k^c = \{z_k^{c,i} | i = 1, \cdots, M\}$ from each camera view to a common view ν using a projection matrix $\mathbf{H}^{c,\nu}$ [17]:

$$z_k^{c,\nu,i} = \mathbf{H}^{c,\nu} z_k^{c,i}, \tag{1}$$

where M is the number of measurements at time k and $z_k^{c,\nu,i}$ is the projection of the measured features from C_c to ν.

The correspondence between feature points is performed using a similarity measure (e.g. Euclidean distance between points or color histogram similarity [22], [23]). When the feature is the occupancy mask of a target in each view, the mask can be projected to obtain an aggregated occupancy on a common view [2]. After correspondence, state estimation is performed using as measurements $Z_k^{\nu,i} = \{z_k^{c,\nu,i} | c \in C_v^i\}$ the features belonging to the same object, where C_v^i is the set of cameras observing target i.

For *trajectory correspondence*, the states $\mathbf{x}_k^{c,i}$ of the object are projected from each view to a common view ν using $\mathbf{H}^{c,\nu}$ [17]:

$$\mathbf{x}_k^{c,\nu,i} = \mathbf{H}^{c,\nu} \mathbf{x}_k^{c,i}. \tag{2}$$

After projection, the tracks $X_k^{c,\nu,i} = \{\mathbf{x}_1^{c,\nu,i}, \cdots, \mathbf{x}_k^{c,\nu,i}\}_{c=1,\cdots,N;i=1,\cdots,M}$ from different cameras are put in correspondence. Figure 1 shows an example of track correspondence used to generate a global track over the network [5].

The projection matrix $\mathbf{H}^{c,\nu}$ itself can be computed manually by selecting control points [24] or automatically through feature point correspondence in overlapping camera networks using the Scale Invariant Feature Transform (SIFT) [2] or 3-D feature points [25]. Track features such as field-of-view lines [26], [27], trajectory correspondence [28] and activity-related correspondence [29] are also used to compute $\mathbf{H}^{c,\nu}$ for overlapping cameras. In non-overlapping cameras, the relative position and orientation of the sensors can be estimated by assuming, for example, that the track of at least one target is available in each camera [30]. A survey on calibration techniques can be found in [31].

The performance of the correspondence generally relies on *synchronization* among cameras [32]. Synchronization can be achieved through a centralized server that distributes time-stamp information [33] or through events such as flashes that are simultaneously visible by several cameras [34]. The use of dedicated hardware may not be cost-effective for certain applications, whereas synchronization through events may not be applicable in wide-area networks. Automatic synchronization methods introduce a temporal shift in the received measurements from multiple cameras during correspondence till optimal correspondence is achieved. These approaches exploits the invariance property of the projective transformations and uses tracks, visual-hull [35] and geometric constraints of line features [36] to rectify the temporal shift between measurements. Any remaining temporal shifts (e.g. between non-overlapping cameras) are handled as uncertainty in the measurements during target-state estimation.

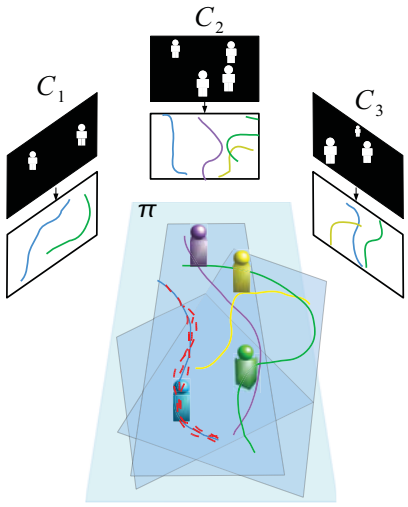


Fig. 1. Trajectory correspondence among multi-camera views.

III. Fusion centers

Fusion centers are nodes in a network that collect data from the cameras within a cluster and perform state estimation. The use of fusion centers favours scalability and reduces the overall communication load by limiting the flow of measurements from nearby nodes within a cluster. Then communication among fusion centers enables the sharing of state estimations among clusters.

Fusion centers can be chosen a priori (fixed fusion centers) or dynamically. Fixed fusion centers are generally used in networks where some nodes have higher processing power and energy supply [7], [37]. Although fixed fusion centers reduce the computational cost for cluster members and increase the lifespan of the overall network, they do not necessarily use the cameras with the best view of a target and hence may generate lower quality observations. To compensate for this limitation, fusion centers can be chosen dynamically, based on trackability measures that evaluate the effectiveness of the features observed by a camera [38]. The features extracted from each view can then be transmitted and compared at a fusion center to decide the next best view. Best-view selection improves target tracking at the cost of an additional computational and communication step due to feature extraction and transmission. The computation overhead can be reduced by using features already extracted for local target tracking, whereas the communication load can be reduced by fusing features at each node to obtain object-level scores [39]. The object-level scores can be aggregated to obtain a view-level score to be transmitted to the fusion center.

Although dynamic fusion centers can select the best view in the cluster, they do not necessarily use the best cameras for tracking. For example, let us consider a target moving from one cluster to another when it is only visible in very

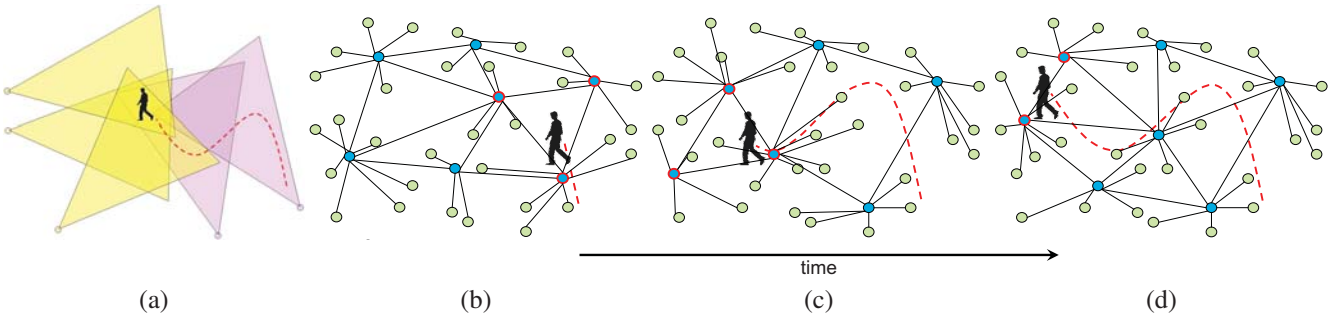


Fig. 2. Multiple dynamic clusters observing a target. (a) Example with two camera clusters identified with yellow and violet triangles that represent the fields of view of the cameras. Note that despite the target is visible in 3 cameras of the yellow cluster and in only 1 camera of the violet cluster, the chosen active cluster could be the violet one as the target was first seen in this cluster. (b-d) Evolution of the camera cluster structure over time as a function of the target position (cameras: green dots; fusion centers: blue dots; active fusion center: red circles).

few cameras within that cluster (Fig. 2(a)). Although there might be cameras belonging to another cluster that can better observe the target, these cameras may not be used as the tracking task has not yet been handed-off to their cluster. In such cases, camera clustering should be adapted on-line and cameras should be added to and removed from the clusters based on target observability [9]. This adaptive cluster-member allocation assigns dynamically fusion centers thus requiring nodes within a cluster to communicate with each other. The clustering procedure can be initiated based on an event such as an object entering a camera view. If an object is detected in several clusters, then these clusters can be used to track a target and may later be merged to form a new cluster. Figure 2(b-d) shows an example of evolution of the cluster structure over time with respect to a moving target.

Although on-line camera clustering introduces additional communication and processing overheads, it allows not only to use the sensors effectively but also to add or remove dynamically cameras to the network, thus improving scalability and robustness against node failures.

IV. Decentralized trackers

Traditional trackers can be extended to multi-camera tracking and used in fusion centers (cluster heads), which receive raw or filtered data from the cameras in a cluster. Notable algorithms such as Graph Matching, Hidden Markov Models, Particle Filters and Kalman Filters (Table I) will be discussed below. A generic flow diagram of decentralized multi-camera tracking is presented in Fig. 3.

A. Graph Matching

Data association via Graph Matching (GM) can be applied on fusion centers where measurements are considered as vertices of a graph [40]. The edge between two vertices, z_a and z_b, is weighted by their similarity $\zeta(z_a, z_b)$. GM computes the vertex disjoint path cover as the sum of weights of all the edges with the aim of finding the maximum-weight path cover over a certain number of consecutive time-steps, which define the depth of the graph [3]. The computational complexity of this algorithm is $O(n^{2.5})$, where n is the number of targets.

Instead of all measurements at each time-step, only the observations at entry and exit points of each camera can be considered as nodes of the graph. The similarity of the targets observed in different cameras is usually measured based on appearance features, such as color. As different camera views are likely to have different illumination conditions, a path smoothness function based on the variance of the observed features can be used [41] to cater for object appearance variations across views.

When cameras have overlapping fields of view, the foreground mask corresponding to targets can be projected from each view onto a top-view plane π, thus generating an occupancy map. GM is then applied on this map [23]. To improve the accuracy of the occupancy map, one can use a multi-level homography generated by moving along the vertical vanishing points and projecting planes parallel to π [2]. The collaboration between the cameras and their fusion center is performed prior to tracking in order to eliminate false detections. The measurements from each view are projected to the fusion center and then back-projected to all the other views for

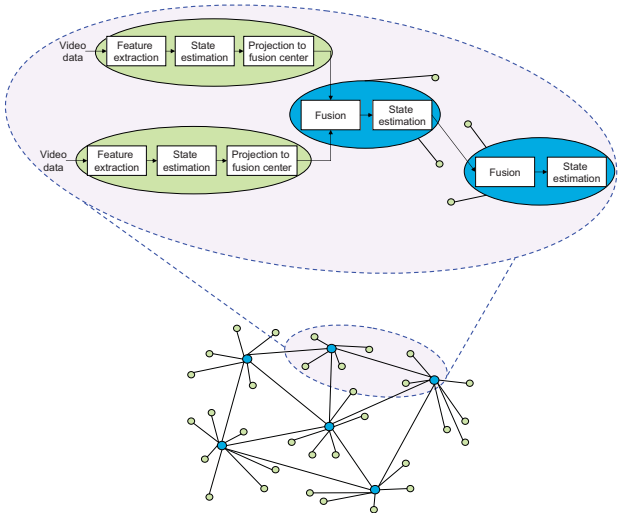


Fig. 3. Data fusion and processing steps in decentralized multi-camera tracking.

TABLE I
MAJOR STEPS OF TARGET STATE ESTIMATION ALGORITHMS

Algorithm	Type	Main steps	
Graph matching	Deterministic	Bipartite graph	Maximum path cover
Hidden Markov Model	Probabilistic	Emission probabilities $p(z_k^{\nu,i}\|\mathbf{x}_k^{\nu,i}=j)$	Viterbi decoding
Particle filter	Probabilistic	Prediction $\mathbf{x}_{k\|k-1}^{\nu,i}=\mathbf{F}_k\mathbf{x}_{k-1\|k-1}^{\nu,i}+\mathbf{v}_k$	Update $\omega_t^{\nu,i,r}\propto\omega_{k-1}^{i}\frac{p(z_k\|\mathbf{x}_k^{\nu,i,r-1})p(\mathbf{x}_k^{\nu,i,r}\|\mathbf{x}_{k-1}^{\nu,i,r})}{q(\mathbf{x}_k^{\nu,i,r}\|\mathbf{x}_{k-1}^{\nu,i,r},z_k)}$
Kalman filter	Probabilistic	Prediction $\mathbf{x}_{k\|k-1}^{\nu,i}=\mathbf{F}_k\mathbf{x}_{k-1\|k-1}^{\nu,i}+\mathbf{v}_k$ $\Omega_{k\|k-1}^{\nu,i}=\mathbf{F}_k\Omega_{k-1\|k-1}^{\nu,i}\mathbf{F}_k^T+\mathbf{Q}_k$	Update $\mathbf{x}_{k\|k}^{\nu,i}=\mathbf{x}_{k\|k-1}^{\nu,i}+\mathbf{K}_k(z_k^{\nu,i}-\mathbf{A}_k\mathbf{x}_{k\|k-1}^{\nu,i})$ $\Omega_{k\|k}^{\nu,i}=(\mathbf{I}-\mathbf{K}_k)\mathbf{A}_k\Omega_{k\|k-1}^{\nu,i}$

validation.

B. Hidden Markov Model

Multi-camera tracking on the occupancy map generated by projecting foreground masks can be performed also based on Hidden Markov Models (HMMs) [42]. Similarly to GM, HMMs compute the target state through graphical modeling, but use Bayesian inference for state estimation under the assumption that the system can be modeled as a Markov process.

Given the observation sequence $Z_k^{\nu,i}$, the set of possible states $\{\mathbf{x}_k^{\nu,i}(u,v)\}_{(u,v)\in(U,V)}$ where (U,V) is the set of all grid locations, the state transition matrix and the initial probability distribution, the likelihood of an observation belonging to the j^{th} state $p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i}=j)$ is computed. Then the state sequence $\mathbf{x}_{1:k}^{\nu,i}$ that maximizes the observation likelihood is obtained using the Viterbi algorithm [42].

Since a HMM assumes a discrete set of states, the search area (occupancy mask) can be divided into a regular grid and the probability of the measurement and the state ending at location a at time k is estimated. The posterior is computed using the Bayesian recursion (see Eq. 6 and Eq. 7) as

$$p(\mathbf{x}_k^{\pi,i}|\Psi_k^{\pi})=p(\Psi_k^{\pi}|\mathbf{x}_k^{\pi,i}=a)\max_K p(\mathbf{x}_k^{\pi,i}=a|\mathbf{x}_{k-1}^{\pi,i}=b)\times \times p(\mathbf{x}_{k-1}^{\pi,i}|\Psi_{k-1}^{\pi}), \quad (3)$$

where $\Psi_k^{\pi}=\{\mathcal{I}_k^{c,\pi}\ |c\in C_v\}$ is the set of projected occupancy masks from the set C_v of cameras observing the target. The likelihood $p(\Psi_k^{\pi}|\mathbf{x}_k^{\pi,i}=a)$ is computed based on the color similarity and $\{a,b\}\in(U,V)$. To estimate the track for each object i, the recursion in Eq. 3 is solved using the Viterbi algorithm. The estimated tracks are then re-projected to each view for validation. Fleuret et al. [42] performed this validation by analyzing the intersections between local estimates on a view with the re-projection from the top view.

The Viterbi decoding makes the HMM-based tracker computationally more efficient than GM-based algorithms. However, a delay is introduced when summing the estimates over several frames.

C. Particle filter

Particle filtering (PF) can be applied in each camera of a cluster to compute local state estimates that are then merged at the fusion center [43]. The tracking problem is solved based on the *state equation*

$$\mathbf{x}_k^{\nu,i}=f_k(\mathbf{x}_{k-1}^{\nu,i},\mathbf{v}_k), \quad (4)$$

and on the *measurement equation*

$$z_k^{\nu,i}=h_k(\mathbf{x}_k^{\nu,i},\mathbf{w}_k), \quad (5)$$

where f_k and h_k are non-linear and time-varying functions. $\{\mathbf{v}_k\}_{k=1,\dots}$ and $\{\mathbf{w}_k\}_{k=1,\dots}$ are assumed to be independent and identically distributed stochastic processes. Given the set of measurements $Z_k^{\nu,i}$ up to time k, the objective is to recursively quantify some degree of belief in the state $\mathbf{x}_k^{\nu,i}$ taking different values, i.e. to estimate the posterior *pdf* $p(\mathbf{x}_k^{\nu,i}|Z_k^{\nu,i})$ using prediction and update. In the *prediction* step, the prior density of the state at time k is obtained from the state estimate at the previous time step using the Chapman-Kolmogorov equation:

$$p(\mathbf{x}_k^{\nu,i}|Z_{k-1}^{\nu,i})=\int p(\mathbf{x}_k^{\nu,i}|\mathbf{x}_{k-1}^{\nu,i})p(\mathbf{x}_{k-1}^{\nu,i}|Z_{k-1}^{\nu,i})d\mathbf{x}_{k-1}, \quad (6)$$

where $p(\mathbf{x}_k^{\nu,i}|\mathbf{x}_{k-1}^{\nu,i})$ is the transition density defined by the target motion and $p(\mathbf{x}_{k-1}^{\nu,i}|Z_{k-1}^{\nu,i})$ is the posterior at time $k-1$. In the *update* step, the posterior density of the state at time k is obtained using the current measurement $z_k^{\nu,i}$. The *update* step is carried out using the measurement at time k by applying the Bayes' rule:

$$p(\mathbf{x}_k^{\nu,i}|Z_k^{\nu,i})=\frac{p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i})p(\mathbf{x}_k^{\nu,i}|Z_{k-1}^{\nu,i})}{\int p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i})p(\mathbf{x}_k^{\nu,i}|Z_{k-1}^{\nu,i})d\mathbf{x}_k}, \quad (7)$$

where $p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i})$ is the likelihood function.

In PF, Sampling Importance Resampling (SIR) [44] is used where a posterior density is represented by a set of particles with their associated weights $\{\omega_k^{\nu,i,r},\mathbf{x}_k^{\nu,i,r}\}$. PF approximates the densities $p(\mathbf{x}_k^{\nu,i}|Z_k^{\nu,i})$ with a sum of Dirac functions centered in $\left\{\mathbf{x}_k^{\nu,i,r}\right\}_{r=1,\cdots,R}$ as

$$p(\mathbf{x}_k^{\nu,i}|Z_k^{\nu,i})\approx\sum_{r=1}^{R}\omega_k^{\nu,i,r}\delta\left(\mathbf{x}_k^{\nu,i}-\mathbf{x}_k^{\nu,i,r}\right), \quad (8)$$

where R is the number of particles. In the prediction step, particles are propagated using the motion model of Eq. 4. In

the update step, the particle weights are updated based on the likelihood as

$$\omega_k^{\nu,i,r} \propto \omega_{k-1}^{\nu,i,r} \frac{p(z_k^{\nu,i}|\mathbf{x}_{k-1}^{\nu,i,r}) p(\mathbf{x}_k^{\nu,i,r}|\mathbf{x}_{k-1}^{\nu,i,r})}{q(\mathbf{x}_k^{\nu,i,r}|\mathbf{x}_{k-1}^{\nu,i,r}, z_k)}, \quad (9)$$

where $q(.)$ is the importance density function.

Transferring particles and their weights across the network may be too expensive in terms of energy consumption and communication overhead. Instead of running one PF at each node, running only one PF on the fusion center reduces energy consumption. Kim and Davis [6] adopted this strategy and projected the object heights across the camera views onto the top view. The intersections of these lines generate a set of points that are used to draw particles during the prediction step (Eq. 6). During the update step, the position represented by each particle is back-projected to the view (Eq. 2) and then associated with the segmented target. The color histogram $\mathcal{H}^{c,i}$ of the target is compared with the reference histogram $\mathcal{H}^{*c,i}$ using an appropriate distance d, based for example on the Bhattacharyya coefficient. The final likelihood $p(z_k^{\pi,i}|\mathbf{x}_{k-1}^{\pi,i})$ for the weight update (Eq. 9) is computed as

$$p(z_k^{\pi,i}|\mathbf{x}_{k-1}^{\pi,i}) = \prod_{c \in C_v} e^{-d(\mathcal{H}^{c,i}, \mathcal{H}^{*c,i})^2}. \quad (10)$$

The communication load can be further reduced by using multiple fusion centers, each using a PF [7]. After the measurements from the cameras are transmitted to their associated fusion center, the particles on each active fusion center can be propagated using the state dynamic model (Eq. 4) and a histogram of the expected measurement values is constructed. The measurements are then transmitted to all the other fusion centers that use the measurements to generate the global estimate of the target state. To reduce data transmission over the network, the product of likelihoods $\prod_r p(z_k|x_k^r)$ performed in the weight update step (Eq. 8, Eq. 9) can be approximated with a parametric model $\mathcal{G}^c(\mathbf{x}_k^c; \phi_k^c)$ [7], where ϕ_k^c are the learned parameters for camera c. In this case, the weight update equation (Eq. 9) is rewritten as

$$\omega_{k|k-1}^{c,i,r} = \frac{\mathcal{G}^c(\mathbf{x}_k^{c,i,r}; \phi_k^c) p(\mathbf{x}_k^{c,i,r}|\mathbf{x}_{k-1}^{c,i,r})}{q(\mathbf{x}_k^{c,i,r}|\mathbf{x}_{k-1}^{c,i,r}, z_k)}. \quad (11)$$

To further reduce the communication cost, Ing and Coates [37] introduced a measurements quantization using the Lloyd-Max algorithm and a vectorization step. The goal of vectorization is to send multiple observations in a single packet to reduce overhead costs. Instead of transmitting the measurements at each time step, they are transmitted only after several time steps have elapsed.

Although quantization, vectorization [37] and parameterization [8] significantly reduce communication costs, PF is still computationally expensive due to the use of multiple particles for state estimation. More economical solutions have been designed, such as for example using a Gaussian Mixture approximation of the particle filtering [8].

D. Gaussian Mixture Particle filter (GMMPF)

Given the high cost of transferring particles, individual PFs can run in parallel in each node and partial results can be updated on each camera sequentially, based on results forwarded from neighbouring nodes and local observations. These local estimates are forwarded to the local fusion center for the estimation of the final output using a dynamic organization of the sensors into clusters based on target trajectories.

Before propagating the information, to reduce computational and communication costs the local sufficient statistics (belief) is approximated by a low-dimensional Gaussian Mixture Model (GMM) as

$$p(\mathbf{x}_k^{\nu,i}|Z_k^{\nu,i}) = \sum_{\theta=1}^{\Theta} \omega_k^{\nu,i,\theta} \mathcal{N}(\mu_k^{\nu,i,\theta}, \sigma_k^{\nu,i,\theta}), \quad (12)$$

where Θ is the number of Gaussians and $(\mu_k^{\nu,i,\theta}, \sigma_k^{\nu,i,\theta})$ are the mean and standard deviation of the Gaussian θ. The posterior distribution estimated by such a distributed PF converges almost surely to the posterior distribution estimated with a centralized Bayesian formulation [8].

E. Track-before-detect Particle filter

To reduce the dependency on the quality of the top-view occupancy map discussed in previous sections, simultaneous detection and tracking can be performed using multi-target track-before-detect [4], implemented using particle filtering (TBDPF). Unlike detection-based tracking that may require hard thresholding, TBDPF performs fusion prior to tracking and the target intensity I_k^{π} within the input signal is included in the target state.

In the prediction step, particles are propagated using a motion model (Eq. 4) and additional particles are generated using an appropriate distribution (e.g. a uniform distribution or a distribution based on normalized measurements). In the update step, the particle weights are recomputed using the likelihood based on target intensity (and not on the color histogram) as

$$p(z_k^{\pi}(u,v)|\mathbf{x}_k^{\pi,r}(u,v)) = \frac{p_{\mathcal{S}+\mathcal{N}}(z_k^{\pi}(u,v)|\mathbf{x}_k^{\pi,r})}{p_{\mathcal{N}}(z_k^{\pi}(u,v))}, \quad (13)$$

where $p_{\mathcal{N}}(.)$ is the *pdf* of the background noise and $p_{\mathcal{S}+\mathcal{N}}(.)$ is the likelihood of the target signal affected by noise. This approach allows us to filter components belonging to noise only, thus enabling tracking without the need of hard thresholding.

The particles are then weighted by computing the product of the likelihood as

$$\omega_{k|k-1}^{\pi,r} = \prod_{u \in w_i(\mathbf{x}_{k|k-1}^{\pi,r})} \prod_{v \in w_j(\mathbf{x}_{k|k-1}^{\pi,r})} p(z_k^{\pi}(u,v)|\mathbf{x}_k^{\pi,r}), \quad (14)$$

where $w_i(.)$ and $w_j(.)$ indicate that only the pixels affected by the target are used in the likelihood computation and are selected by using a fixed-size window. TBDPF requires the transmission of the bounding box of each target or of all the pixels defining the target area. For n targets, the complexity of TBDPF is $O(n^3)$.

F. Kalman Filter

When the use of Monte Carlo methods [7], [37] is too expensive in terms of computation and communication costs, approaches based on the Kalman Filter (KF) are used. The complexity of KF for n targets is $O(n)$ [45]. KF is applicable to linear Gaussian models, whereas for non-linear non-Gaussian models the Extended Kalman filter (EKF) can be used [9], [46]. EKF assumes local linearization and approximates the posterior to be Gaussian.

In the prediction step of the KF, the state and covariance Ω_k are estimated as

$$\mathbf{x}_{k|k-1}^{\nu,i} = \mathbf{F}_k \mathbf{x}_{k-1|k-1}^{\nu,i} + \mathbf{v}_k, \tag{15}$$

$$\Omega_{k|k-1}^{\nu,i} = \mathbf{F}_k \Omega_{k-1|k-1}^{\nu,i} \mathbf{F}_k^T + \mathbf{Q}_k, \tag{16}$$

where $\mathbf{Q}_k$ is the state noise covariance. In the update step, given the observation $z_k^{\nu,i}$, the posterior density of the state and covariance are obtained as

$$\mathbf{x}_{k|k}^{\nu,i} = \mathbf{x}_{k|k-1}^{\nu,i} + \mathbf{K}_k (z_k^{\nu,i} - \mathbf{A}_k \mathbf{x}_{k|k-1}^{\nu,i}), \tag{17}$$

$$\Omega_{k|k}^{\nu,i} = (\mathbf{I} - \mathbf{K}_k) \mathbf{A}_k \Omega_{k|k-1}^{\nu,i}, \tag{18}$$

where $\mathbf{K}_k$ is the Kalman gain, $\mathbf{A}_k$ is the measurement transition matrix and $\mathbf{I}$ is the identity matrix.

The transmission over the network of the matrices (although small) can introduce a communication load for high-dimensional target states. Medeiros et al. [9] reduce the communication cost by exploiting the sparsity of the Jacobian matrix of $\mathbf{F}$ and $\mathbf{A}$ in Eq. 15 and Eq. 17. Moreover, the Sign of Innovation Kalman filter (SOI-KF) can be used to further reduce these costs [47]. Only the sign of the error is transmitted (instead of the error covariance matrix), thus reducing the communication overhead. However, the signs of innovations have to be correctly received by each node of the network, thus reducing scalability. SOI-KF can be implemented using an observation-transmission algorithm where a scheduler selects nodes that collect the observation and executes the SOI-KF [47].

V. Distributed trackers

Unlike decentralized trackers, distributed trackers have no specific local fusion centers: each node fuses its estimates with information received from its neighbours and projects the updated estimates to the next neighbour until the last node is reached or a desired accuracy is achieved. State estimation for distributed trackers can be based on Particle Filters [11] or on Kalman Consensus Filters [10], [12]–[15]. A generic flow diagram of distributed multi-camera trackers is shown in Fig. 4.

A. Particle filter

Distributed trackers do not use occupancy masks due to their high communication load. Instead, the bounding box of a target in each view, represented for example by a 2D Gaussian, can be projected to the neighbouring node. The overlap of these Gaussians from multiple neighbours generates the target

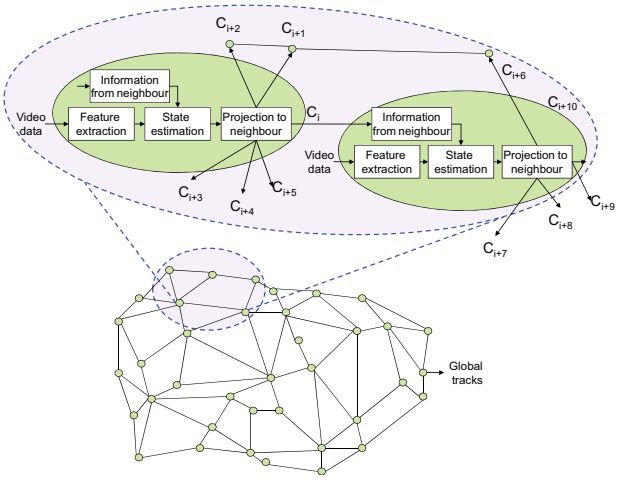


Fig. 4. Data fusion and processing steps in distributed multi-camera tracking.

position estimation with the uncertainty of the most accurate measurement. This process of fusion and projection continues until tracking is performed at a sink node.

Xue and Sheng [11] further improve state estimation using PF for distributed tracking with a state transition $p(\mathbf{x}_k^{\nu,i}|\mathbf{x}_{k-1}^{\nu,i})$ based on the Interacting Multi-Model (IMM) filter [44]. IMM allows the use of multiple transition models λ_k^i, for which the prediction and update equations (Eq. 6 and Eq. 7) can be rewritten as

$$p(\mathbf{x}_k^{\nu,i}, \lambda_k^i = h|Z_{k-1}^{\nu,i}) = \sum_g \gamma_{g,h} \int p(\mathbf{x}_k^{\nu,i}|\mathbf{x}_{k-1}^{\nu,i}, \lambda_k^i = h) \times \\ \times p(\mathbf{x}_{k-1}^{\nu,i}, \lambda_{k-1}^i = g|Z_{k-1}^{\nu,i}) d\mathbf{x}_{k-1}, \tag{19}$$

$$p(\mathbf{x}_k^{\nu,i}, \lambda_k^i = h|Z_k^{\nu,i}) = \\ = \frac{p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i}, \lambda_k^i = h) p(\mathbf{x}_k^{\nu,i}, \lambda_k^i = h|Z_{k-1}^{\nu,i})}{\sum_h \int p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i}, \lambda_k^i = h) p(\mathbf{x}_k^{\nu,i}, \lambda_k^i = h|Z_{k-1}^{\nu,i}) d\mathbf{x}_k}, \tag{20}$$

where g and h are the state transition model of the i^{th} target at time $k-1$ and k, respectively. The weighted estimate can be obtained as [44]

$$\omega_k^{\nu,i,r} = p(z_k^{\nu,i}|\mathbf{x}_k^{\nu,i,r}, \lambda_k^i) p(\mathbf{x}_k^{\nu,i,r}|\mathbf{x}_{k-1}^{\nu,i,r}, \lambda_k^i), \tag{21}$$

where the parameter λ_k^i is generally modeled as a first-order Markov model. Due to multiple prediction and update steps, the use of IMM increases the computational complexity of the tracker. The execution time can be improved by training a Radial Basis Function Network (RBFN) for a faster convergence [11].

B. Kalman Consensus Filter

The Kalman Consensus Filter (KCF) performs tracking locally in each camera using information from neighbouring nodes. As each camera may have an inconsistent information, the goal of the filter is to achieve consensus over time by exchanging data [48].

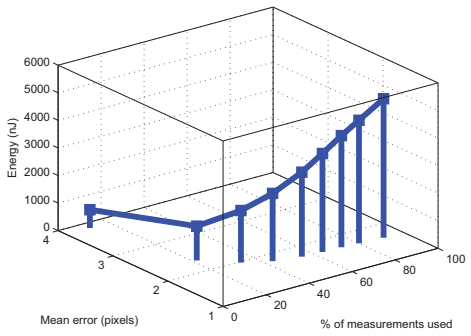


Fig. 5. Communication cost vs. accuracy while increasing the number of measurements for KCF. Note that the rate of accuracy improvement with respect to the communication cost is much reduced after adding more than 50% of the total measurements.

The cameras transmit the measurement consensus $\tilde{z}_k^i$ and the inverse-covariance consensus $\tilde{\mathbf{R}}$ [12] to the neighbouring node C_j. For each target i, the two consensus terms are calculated using the measurement matrix and the measurement noise covariance $\mathbf{R}$ as

$$\tilde{z}_k^c = \mathbf{A}_K^{c\,T} \mathbf{R}_k^{c\,-1} z_k^c \tag{22}$$

$$\tilde{\mathbf{R}}^c = \mathbf{A}_K^{c\,T} \mathbf{R}_k^{c\,-1} \mathbf{A}_k^i. \tag{23}$$

These consensus terms are received along with the estimated state by the neighbouring node where the consensus terms from all the neighbours are fused as

$$\tilde{\mathbf{z}}_k^{C_j} = \sum_{c \in C_v} \tilde{z}_k^c \tag{24a}$$

$$\mathbf{S}^{C_j} = \sum_{c \in C_v} \tilde{\mathbf{R}}^c. \tag{24b}$$

Because we can compute the average inverse covariance $\mathbf{S}^{C_j}$ and the average measurement at each camera in the network, the state estimate for KCF is obtained at each camera by rewriting the update of the Kalman filter (Eq. 17) as [12]

$$\mathbf{x}_{k|k}^{C_j} = \mathbf{x}_{k|k-1}^{C_j} + \mathbf{K}_k^{C_j} (\tilde{\mathbf{z}}_k^{C_j} - \mathbf{S}_k^{C_j} \mathbf{x}_{k|k-1}^{C_j}), \tag{25}$$

$$\bar{\mathbf{x}}_{k|k}^{C_j} = \mathbf{F}_k \hat{\mathbf{x}}_{k|k}^{C_j}, \tag{26}$$

$$\Omega_{k|k}^{C_j} = \mathbf{F}_k \mathbf{K}_k^{C_j} \mathbf{F}_k^T + \mathbf{Q}_k, \tag{27}$$

where $\mathbf{K}_k^{C_j} = (\Omega_{k|k-1}^{-1} + \mathbf{S}_k^{C_j\,-1})^{-1}$ is the Kalman gain. Note that the gain $\mathbf{K}_k^{C_j}$ in KCF is of dimension $\tau \times \tau$ instead of $\tau \times \tau \times N$ as in the conventional (decentralized) Kalman filter [12]. In the decentralized approach, the target state $\mathbf{x}_k^{\pi,i}$ was in fact estimated using a $N \times \tau$ dimensional measurement, where τ is the measurement dimensions. In case of distributed tracking, the same state estimate can be obtained at a sink node where the consensus is achieved by using just a τ-dimensional measurement. This makes KCF computationally feasible for distributed networks.

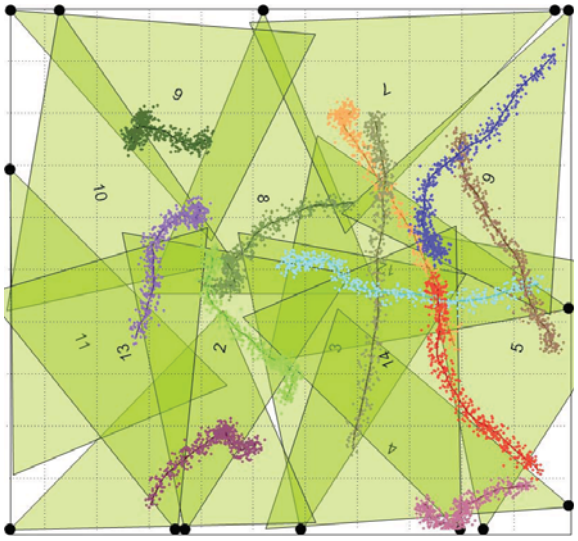


Fig. 6. Multi-camera network with $N = 14$ and $T = 12$ targets. Tracks are shown with positions corrupted by various noise levels.

The accuracy of state estimation improves with the increase of shared information, at the cost of additional energy dissipation. Figure 5 shows a trade-off between accuracy, amount of measurements and the communication cost for KCF, using the data shown in Fig. 6. Although the communication cost increases at the same rate as the amount of measurements, it is important to notice that over 50% of the estimation errors are removed by just increasing the measurements from 0% to 33%. The remaining 77% of the measurements only contributes an 8% improvement in accuracy at the same cost.

A major challenge for KCF-based approaches [10], [15] is the definition of a strategy for the appropriate selection of sink nodes, that can be defined statically or selected dynamically based on target positions.

VI. Quantifying Communication and Computation Costs

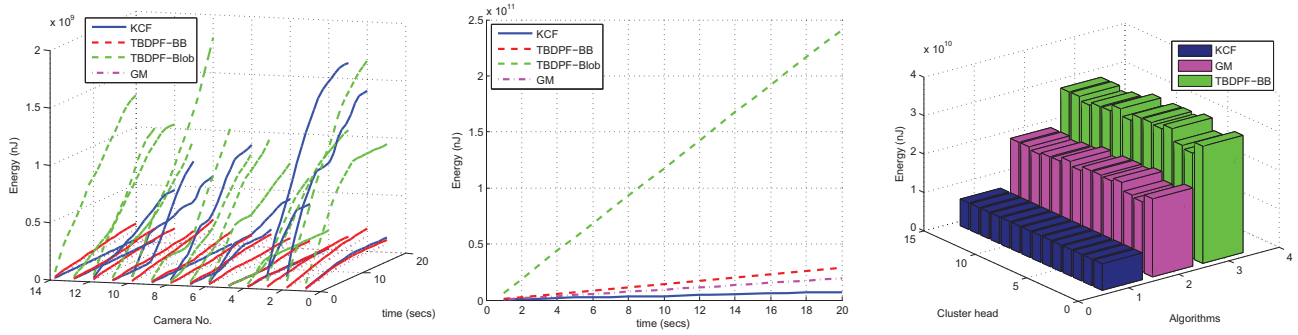
A. Modeling costs

Sharing data among cameras uses a certain amount of energy per bit when messages or p-bit packets are transmitted. The communication cost depends on the packet size per target, the number of targets and the number of messages transmitted and received. Let for simplicity each message reach its destination using a single hop. For wireless cameras, we can use the *Energy Consumption Models* for the radio module energy dissipation for communication and computation [49]. The energy $E_T(.)$ that is necessary for transmitting a p-bit packet to distance d is

$$E_T(p, d) = E_e p + \epsilon_a p d^2, \tag{28}$$

where E_e is the electrical energy in Joules/bit and ϵ_a is the power amplification needed to obtain an acceptable signal strength at the receiver. Using the same model, the energy $E_R(.)$ consumed in receiving the p-bit packet is

$$E_R(p) = E_e p. \tag{29}$$



(a) (b) (c)

Fig. 7. Comparison of communication costs for KCF, TBDPF (TBDPF-BB: TBDPF using the bounding box of targets only; TBDPF-Blob: TBDPF using all the points defining a target) and GM. (a) Communication cost over time for each camera in the network. (b) Communication cost over the whole network. (c) Communication cost when selecting one camera as fusion center.

In addition to the communication costs, each computation cycle in a node has an associated cost in terms of energy consumption. This energy E_c depends on the average capacitance switched per cycle C and the number of cycles L required to complete a task, and is defined as

$$E_c = LCV_s^2, \tag{30}$$

where V_s is the supply voltage. As a certain amount of energy is leaked in electrical components, the cost $E_l(.)$ associated to these leakages can be computed as

$$E_l(V_s, f) = V_s \left(I_o e^{\frac{V_s}{\eta V_T}} \right) \left(\frac{L}{f} \right), \tag{31}$$

where V_T is the thermal power, f is the clock speed of the processor and η is a CPU-dependent parameter.

The use of these measures allows us to understand if the energy across the nodes is consumed evenly, which is an important property as the lifetime of a camera network is equal to the shortest lifetime of individual nodes [11].

B. Comparative example

In order to appreciate the differences among the algorithms, let us consider a network of 14 cameras that observe 12 targets that move with varying velocities on a plane and follow a random walk (Fig. 6). We compare the following trackers: Graph Matching (GM) [3], track-before-detect particle filtering (TBDPF) [4] and Kalman Consensus filtering (KCF) [15]. The generated tracks are corrupted with different levels of Gaussian noise, with $\sigma = \{1, 1.2, 1.4, 1.6, 1.8, 2.0\}$, in order to quantify the robustness of the algorithms to each noise level. The parameters of all the algorithms are kept the same for all noise levels. For simplicity, but without loss of generality, we consider rectified top-view cameras and discard distortions that are common to all trackers and might be introduced by clutter, illumination changes, occlusions, and synchronization issues. We conduct three experiments to measure communication cost, computation cost and tracking accuracy of each algorithm.

The communication cost depends on the packet size per target, the number of targets and the number of messages transmitted and received. We consider a simple simulation environment where the packet size for each algorithm depends only on the size of the state vector and it is equal to the number of bytes required to store the data. We also assume that the information for each target at each time-step can be associated to a single packet that is transferred using one message in a single hop. Let the position and velocity of a target be 32-bit floating point numbers and the size be 16-bit integer. KCF transmits a 4D state vector and the 4×4 covariance matrix from one camera to the next to obtain the consensus. Assuming that each target information is sent as a single packet, each KCF packet requires $32 \times (4 + 4 \times 4) = 640$ bits. GM needs the target position (x, y) from each camera to be available at the fusion node. Thus the packet size per target is 64 bits. Unlike KCF and GM, TBDPF requires transmitting at least the bounding box for each target or all the pixels belonging to the target. In the former case the packet size is 96 bits per target, whereas in the latter case an average of 800 bits per target are transmitted if the size of each target is 5×10. We do not consider in this comparison the additional bits associated with each message, as they are common to all the algorithms.

The communication and computation costs are estimated for each camera as well as for the entire network, including the processing at the fusion centers, by varying the number of targets appearing in each camera view. These costs are estimated using data affected by the same amount of noise for all the algorithms. The position and field of view of each camera and the position of the fusion centers are also kept constant. Regarding the estimation of L, as it would be difficult to obtain a fair comparison of the actual number of floating point operations per second because this would require equally efficient implementations for all the algorithms, we compute the number of targets n at each respective view (camera view for KCF and estimated top-view for GM and TBDPF) to be used in the respective complexity formula. Therefore, for each time step k, $O(n^{2.5})$ for GM, $O(n^3)$ for TBDPF, and $O(n(n+1))$ for KCF with a nearest-neighbour (NN) track-to-measurement association. In the example we assume that each sensor is a μ AMPS node [49] with the following parameters: $C = 0.67$nF, $V_s = 0.85$V, $f = 74$MHz, $I_0 = 1.196$mA,

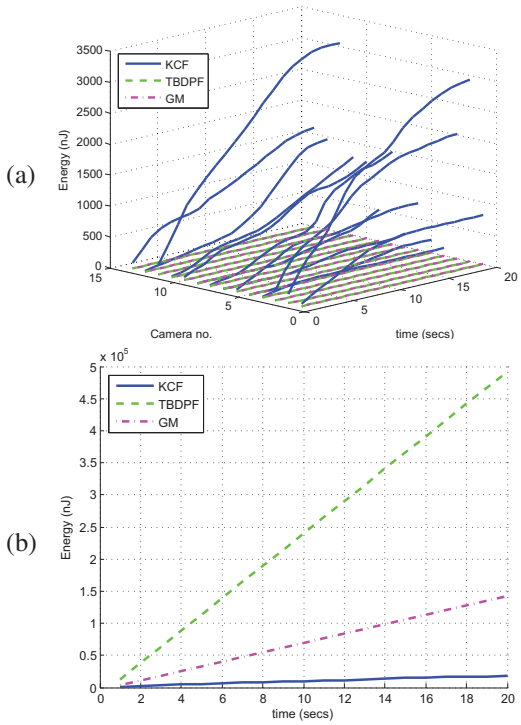


Fig. 8. Computation cost for KCF, TBDPF and GM. (a) Computation cost over time for each camera in the network. (b) Computation cost over the whole network (including fusion centers).

$\eta = 21.26$, $V_T = 1$V, $E_e = 50$ nJ; $\epsilon_a = 0.1$ nJ. Finally, we do not consider in the comparison the data acquisition and pre-processing steps that are common to all the algorithms.

C. Discussion

Figure 7 shows the *communication cost* both at camera level and at network level. Although KCF has a higher communication cost *per camera* (as it requires to transmit not only the state but also the covariance matrix associated to each target), it requires less energy for communication over the *whole network* (Fig. 7(b)). In fact, as a distributed algorithm, KCF communicates only with close-by neighbours and therefore uses less energy for signal amplification. In a decentralized algorithm, instead, the fusion center is close to some nodes but farther from other nodes, thus requiring a stronger amplification for transmission. The energy efficiency of KCF can also be improved by exploiting the sparsity of the matrix to be transmitted [9]. The cost for TBDPF can be reduced if only the bounding box or contour information is transmitted, instead of all the points (see Fig. 7(a-b), TBDPF-BB vs. TBDPF-Blob).

In general, the cameras observing targets more frequently and those farther away from the fusion center will incur a higher energy consumption (e.g. C_2, C_3 and C_{10} in Fig. 7(a)). To ensure comparable energy usage among cameras (i.e. to increase the lifespan of the network), uniform energy consumption can be imposed [11]. Moreover, energy dissipation can be minimized in decentralized approaches by selecting as cluster center the closest node that also shares view with most of the nodes in the cluster (Fig. 7(c)). For example, C_9 would be the worst choice as fusion center, whereas C_3 would be the best choice to minimize energy dissipation.

Figure 8 compares the *computational cost* for the three algorithms. Given a fixed number of targets, the cumulative energy consumption for GM and TBDPF grows linearly over time. Although the complexity of GM and KCF are very similar, KCF has the lowest network-wide energy consumption as it distributes the processing to the cameras observing a target. Despite this lower overall energy cost and despite requiring fewer parameters to be tuned compared to TBDPF, the performance of KCF degrades if individual observations are less accurate, as the distributed algorithm needs to calculate the average consensus. Moreover, the NN-based track-to-measurement association used by KCF is only applicable to simple scenarios and a more sophisticated algorithm, such as the Hungarian or the Joint Probabilistic data association [21] should be employed in more complex scenarios, thus increasing the energy consumption of a KCF-based algorithm compared to a GM-based algorithm. Note that since consensus-based algorithms such as KCF rely on individual sensors to compute their own estimates, they can employ other methods (e.g. an Extended Kalman filter) to improve performance in challenging scenarios.

The *accuracy* of the tracking results is evaluated by measuring the Euclidean distance between the estimated and the ground-truth tracks. To measure the accuracy, we use data with varying level of Gaussian noise while keeping the same amount of computation and communication. The mean and standard deviation of the error for up to 500 frames are shown in Fig. 9. It is possible to notice that the accuracy of GM decreases (both in terms of mean error and its standard deviation) with the increase of the noise level[3], whereas the estimation errors for KCF and TBDPF increase at a slower rate with the increase of the amount of noise. Moreover, as the noise increases, the number of iterations for KCF to reach consensus also increases [12].

In general, TBDPF performs state estimation with higher accuracy, but it is less suitable for networks where energy dissipation is to be kept to a minimum. KCF is an energy-efficient alternative, but its energy efficiency is reduced with the increase of noise in the observations. PF-based approaches are therefore likely to outperform KCF in the presence of other distortions due to noise, clutter and occlusions. To address these limitations, closed-form solutions such as Gaussian mixture approximations of Monte Carlo methods could be used or, instead of performing association at each time-step, associations could be enabled only at field-of-view lines for consistent multi-camera labeling. Table II summarizes the characteristics of the three algorithms analyzed in this section.

[3]Note that GM only solves the correspondence problem and does not smooth the measurements.

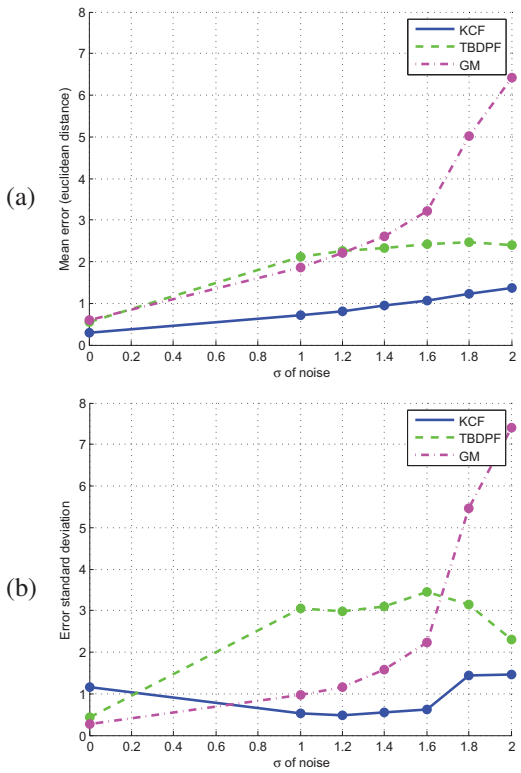


Fig. 9. Track estimation errors for KCF, TBDPF and GM as a function of the measurement noise. (a) Mean error. (b) Standard deviation of the error.

VII. Summary

We discussed emerging multi-camera tracking algorithms that find their roots in signal processing, wireless sensor networks and computer vision. Based on how cameras share estimates and fuse information, we classified these trackers as distributed, decentralized and centralized algorithms. We also highlighted the challenges to be addressed in the design of decentralized and distributed tracking algorithms. In particular, we showed how the constraints derived from the topology of the networks and the nature of the task have favoured so far decentralized architectures with multiple local fusion centers. Due to the availability of fewer fusion centers compared to distributed algorithms, decentralized algorithms can share larger amounts of data (e.g. occupancy maps) and can back-project estimates among views and fusion centers to validate results. Distributed tracking uses algorithms that can operate with smaller amounts of data at any particular node and obtain state estimates through iterative fusion.

Despite recent advances, there are important issues to be addressed in order to achieve efficient multi-target multi-camera tracking. Current algorithms either assume the track-to-measurement association information to be available for the tracker or operate on a small (known) number of targets. Algorithms performing track-to-measurement association for a time-varying number of targets with higher accuracy usually incur much higher costs, whose reduction is an important open problem to be addressed in multi-camera networks.

TABLE II
Main characteristics of Kalman Consensus Filter (KCF), track-before-detect particle filter (TBDPF) and graph matching (GM) for multi-camera tracking.

		Algorithm		
		GM	**TBDPF**	**KCF**
Type	Deterministic	✓		
	Probabilistic		✓	✓
	Multi-target	✓	✓	✓[4]
Arch.	Decentralized	✓	✓	
	Distributed			✓
Feature	Robustness to noise		✓	✓
	Low communication cost	✓		✓
	Low computational cost			✓
	Low overall energy consumption			✓

VIII. Acknowledgment

This work was supported in part by the EU, under the FP7 project APIDIS (ICT-216023).

References

[1] P. Baran, "On distributed communications networks," *IEEE Trans. Communications Systems*, vol. 12, no. 1, September 1964.

[2] S. Khan and M. Shah, "Tracking multiple occluding people by localizing on multiple scene planes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 505–519, March 2009.

[3] M. Taj, E. Maggio, and A. Cavallaro, "Multi-feature graph-based object tracking," in *CLEAR, Springer LNCS*, vol. 4122, Southampton, UK, April 2006.

[4] M. Taj and A. Cavallaro, "Multi-camera track-before-detect," in *Proc. of ACM/IEEE Int. Conf. on Distributed Smart Cameras*, Como, IT, August 2009.

[5] N. Anjum and A. Cavallaro, "Trajectory association and fusion across partially overlapping cameras," in *Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, Genova, IT, September 2009.

[6] K. Kim and L. S. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering," in *Proc. of the European Conf. on Computer Vision*, Graz, AT, May 2006.

[7] M. Coates, "Distributed particle filters for sensor networks," in *Proc. of Int. Symp. Information Processing in Sensor Networks*, Berkeley, CA, USA, April 2004, pp. 99–107.

[8] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Proc. of Int. Symp. Information Processing in Sensor Networks*, Los Angeles, CA, USA, April 2005, pp. 181–188.

[9] H. Medeiros, J. Park, and A. Kak, "Distributed object tracking using a cluster-based Kalman filter in wireless camera networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 448–463, August 2008.

[10] R. Olfati-Saber and N. Sandell, "Distributed tracking in sensor networks with limited sensing range," in *Proc. of the American Control Conference*, Seattle, WA, USA, June 2008, pp. 3157–3162.

[11] X. Wang and S. Wang, "Collaborative signal processing for target tracking in distributed wireless sensor networks," *Journal of Parallel Distributed Computing*, vol. 67, no. 5, pp. 501–515, May 2007.

[12] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *Proc. of the IEEE Int. Conf. on Decision and Control - European Control Conference*, Seville, ES, December 2005.

[13] X. Wang, S. Wang, D. Bi, and J. Ma, "Distributed peer-to-peer target tracking in wireless sensor," *Sensors*, vol. 7, pp. 1001–1027, 2007.

[14] C. H. Yu, K. H. Lee, J. W. Choi, and Y. B. Seo, "Distributed single target tracking in underwater wireless sensor networks," in *Conf. on SICE*, August 2008, pp. 1351–1356.

[4]Multi-target tracking with KCF requires an external data association method whose performance may affect computational and energy efficiency as well as robustness to noise.

[15] B. Song, A. Kamal, C. Soto, C. Ding, J. Farrell, and A. Roy-Chowdhury, "Tracking and activity recognition through consensus in distributed camera networks," *IEEE Trans. on Image Processing*, vol. 19, no. 10, pp. 2564 –2579, October 2010.

[16] Y. Wang, S. Velipasalar, and M. Casares, "Cooperative object tracking and composite event detection with wireless embedded smart cameras," *IEEE Trans. on Image Processing*, vol. 19, no. 10, pp. 2614–2633, October 2010.

[17] M. Taj and A. Cavallaro, *Computer Vision: Detection, Recognition and Reconstruction*. Springer Verlag GmbH, 2010, ch. 8: Multi-view multi-object detection and tracking, pp. 263–280.

[18] H. Aghajan and A. Cavallaro, *Multi-camera Networks: principles and applications*. Burlington, MA, USA: Elsevier, 2009.

[19] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, December 2006.

[20] D. Smith and S. Singh, "Approaches to multisensor data fusion in target tracking: A survey," *IEEE Trans. on Knowledge and Data Engineering*, vol. 18, no. 12, pp. 1696–1710, 2006.

[21] J. Liu, M. Chu, and J. Reich, "Multitarget tracking in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 36–46, May 2007.

[22] F. Porikli, "Inter-camera color calibration by correlation model function," in *Proc. of IEEE Int. Conf. on Image Processing*, vol. 2, Barcelona, ES, September 2003.

[23] D. Delannay, N. Danhier, and C. D. Vleeschouwer, "Detection and recognition of sports (wo)man from multiple views," in *Proc. of ACM/IEEE Int. Conf. on Distributed Smart Cameras*, Como, IT, August 2009.

[24] A. Zisserman and R. I. Hartley, *Multiple View Geometry in Computer Vision*. Cambridge University Press, U.K, 2004.

[25] J. Kassebaum, N. Bulusu, and W.-C. Feng, "3-D target-based distributed smart camera network localization," *IEEE Trans. on Image Processing*, vol. 19, no. 10, pp. 2530 –2539, October 2010.

[26] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355–1360, October 2003.

[27] E. Lobaton, R. Vasudevan, R. Bajcsy, and S. Sastry, "A distributed topological camera network representation for tracking applications," *IEEE Trans. on Image Processing*, vol. 19, no. 10, pp. 2516 –2529, October 2010.

[28] C. Stauffer and K. Tieu, "Automated multi-camera planar tracking correspondence modeling," in *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Madison, WI, USA, July 2003.

[29] E. Ermis, P. Clarot, P.-M. Jodoin, and V. Saligrama, "Activity based matching in distributed camera networks," *IEEE Trans. on Image Processing*, vol. 19, no. 10, pp. 2595–2613, October 2010.

[30] N. Anjum, M. Taj, and A. Cavallaro, "Relative position estimation of non-overlapping cameras," in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Honolulu, HI, USA, April 2007.

[31] R. J. Radke, *Handbook of Ambient Intelligence and Smart Environments*. Springer, 2010, ch. A survey of distributed computer vision algorithms.

[32] A. Whitehead, R. Laganiere, and P. Bose, "Temporal synchronization of video sequences in theory and in practice," in *IEEE Int. Workshop on Motion and Video Computing*, vol. 2, Breckenridge, CO, USA, 2005, pp. 132–137.

[33] J. Lichtenauer, M. Valstar, J. Shen, and M. Pantic, "Cost-effective solution to synchronized audio-visual capture using multiple sensors," in *Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, Genova, IT, September 2009.

[34] P. Shrestha, M. Barbieri, H. Weda, and D. Sekulovski, "Synchronization of multiple video recordings based on still camera flashes," in *Proc. of ACM Int. Conf. on Multimedia*, Santa Barbara, CA, USA, 2006, pp. 137–140.

[35] S. N. Sinha and M. Pollefeys, "Visual-hull reconstruction from uncalibrated and unsynchronized video streams," in *Proc. of the Int. Symp. on 3D Data Processing, Visualization, and Transmission*, Washington, DC, USA, 2004, pp. 349–356.

[36] C. Lei and Y. Yang, "Tri-focal tensor-based multiple video synchronization with subframe optimization," *IEEE Trans. on Image Processing*, vol. 15, no. 9, pp. 2473–2480, September 2006.

[37] G. Ing and M. Coates, "Parallel particle filters for tracking in wireless sensor networks," in *Workshop on Signal Processing Advances in Wireless Communications*, NY, USA, June 2005, pp. 935–939.

[38] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan, "Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons," in *Proc. of ACM/IEEE Int. Conf. on Distributed Smart Cameras*, Vienna, AT, September 2007.

[39] F. Daniyal, M. Taj, and A. Cavallaro, "Content and task-based view selection from multiple video streams," *Springer Journal of Multimedia Tools and Applications*, vol. 2–3, no. 46, pp. 235–258, January 2010.

[40] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 51–65, January 2005.

[41] B. Song and A. Roy-Chowdhury, "Robust tracking in a camera network: A multi-objective optimization framework," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 582–596, August 2008.

[42] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 267–282, February 2008.

[43] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Proc. of Conf. on Uncertainty in AI*, Acapulco, MX, August 2003.

[44] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. London, UK: Artech House, 2004.

[45] M. S. Grewal and A. P. Andrews, *Kalman filtering: theory and practice using MATLAB*. Wiley Publishing, Inc., 2008.

[46] J. Yoder, H. Medeiros, J. Park, and A. Kak, "Cluster-based distributed face tracking in camera networks," *IEEE Trans. on Image Processing*, vol. 19, no. 10, pp. 2551 –2563, October 2010.

[47] A. Ribeiro, G. Giannakis, and S. Roumeliotis, "SOI-KF: Distributed Kalman filtering with low-cost communications using the sign of innovations," *IEEE Trans. on Signal Processing*, vol. 54, no. 12, pp. 4782–4795, December 2006.

[48] M. Alighanbari and J. How, "An unbiased Kalman consensus algorithm," in *Proc. of the American Control Conference*, Minneapolis, MN, USA, June 2006.

[49] A. Wang and A. Chandrakasan, "Energy-efficient dsps for wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 4, pp. 68–78, July 2002.