Distributed Sensing and Processing for Multi-Camera Networks

Aswin C. Sankaranarayanan, *Member, IEEE,* Rama Chellappa, *Fellow, IEEE,*
and Richard G. Baraniuk, *Fellow, IEEE*

Abstract

Sensor networks with large numbers of cameras are becoming increasingly prevalent in a wide range of applications, including video conferencing, motion capture, surveillance, and clinical diagnostics. In this chapter, we identify some of the fundamental challenges in designing such systems. We focus on three aspects: robust statistical inference, computationally efficiency, and opportunistic and parsimonious sensing. We show that the geometric constraints induced by the imaging process are extremely useful for identifying and designing optimal estimators for object detection and tracking tasks. We also derive pipelined and parallelized implementations of popular tools used for statistical inference in non-linear systems, of which multi-camera systems are examples. Finally, we highlight the use of the emerging theory of compressive sensing in reducing the amount of data sensed and communicated by a camera network.

I. Introduction

Over the past decade, camera networks have become increasingly prevalent in a wide range of applications [1] for a number of different perspectives. These systems are deployed with the objective of capturing, analyzing, and storing high bandwidth video data. Such an infrastructure, when provided with the necessary computing power, is capable of delivering highly sophisticated services that can potentially improve quality of life. Despite recent advances in computer vision, however, many challenges remain before these systems will be truly autonomous and able to detect, track and analyze behaviors in real-time. For example, real-time scene and behavior analysis on visual sensor networks is the first step towards effective scene understanding. Large visual networks also sense a deluge of data which significantly increases the processing and communication costs in a network. In order to

A. C. Sankaranarayanan and R. G. Baraniuk are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX. e-mail: {saswin, richb}@rice.edu

R. Chellappa is with the Center for Automation Research and the Department of Electrical and Computer Engineering at the University of Maryland, College Park, MD. e-mail: rama@cfar.umd.edu

This research was partially supported by the Office of Naval Research under the contracts N00014-09-1-1162 and N00014-07-1-0936, the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-09-1-0383, and the AFOSR under the contracts FA9550-09-1-0432 and FA9550-07-1-0301.

The authors thank Prof. Volkan Cevher, Prof. Ankur Srivastava, Dr. Ashok Veeraraghavan, Dr. Marco Duarte and Mr. Dikpal Reddy for valuable discussions and collaborations.

effectively mine real-time data from such large camera networks, there is an immediate need for computationally efficient and robust algorithms for video analytics from as as little data as possible.

Distributed video processing in a network of smart cameras raises research challenges that can be broadly clustered into three areas:

- **Robust statistical inference:** How does one fuse the information extracted at the individual camera nodes in order to solve detection, tracking, and recognition tasks in a visual sensor network? Invariably, these fusion algorithms involve the interplay between geometric constraints that arise from models of the scene and the imaging process. Inference in such a setting is about designing appropriate statistical estimation techniques that can contend with varying sources of error under the geometric constraints introduced by the imaging process.
- **Computationally efficient and distributed algorithms:** In addition to the distributed sensing aspect in a multi-camera network, the availability of computing resources at each sensing node (such as in a smart camera) raises the possibility of distributed algorithms that minimize communication and energy costs as well as provide robustness to node failures. However, such a framework requires a complete reworking of the commonly used tools and algorithms. In particular, statistical inference techniques that are widely used across many application domains need to address not just the distributed sensing but also the distributed processing nature of the camera network.
- **Opportunistic and parsimonious sensing:** One of the key challenges in the deployment of large camera networks is the processing and storage of the vast amounts of data produced. A fundamental reason for this "data deluge" is that traditional cameras do not exploit the redundancy in the signal at sensing. For many applications, this is extremely wasteful. Toward this end, it is important to design novel sensors and sensing protocols that sense at the information rate of the scene under view. It is also important that the sensing process be adaptive and tuned to the application.

There are numerous applications of distributed visual sensing algorithms. In this chapter, we concentrate on the problems of distributed detection and tracking. These form integral subsystems of any robust distributed visual sensing network. The number of cameras connected in the distributed network can vary greatly: from a few (less than ten) cameras for video conferencing to tens of cameras to monitor a building, to hundreds of cameras connected in a traffic monitoring network, and potentially thousands over a city (such as London or New York). The specific challenges encountered in each of these applications vary with the number of cameras that are connected in the network. Nevertheless, some of the basic principles of algorithm design and optimization remain the same in all these scenarios.

This chapter is organized as follows. We introduce the distributed tracking problem in Section II. In Section III, we discuss implementations of statistical inference algorithms that are inherently distributed and parallelized. Finally, in Section IV, we discuss novel sensing strategies and associated inference algorithms that address the data deluge problem.

II. Statistical inference for multi-camera tracking

In this section, we describe the problems of detection and tracking fin multi-camera networks. Our approach relies heavily on the geometric constraints specific to multi-camera networks. We introduce one such constraint called the homography, and demonstrate the use of homography for multi-view detection and tracking.

A. Homography

Central projection is the fundamental principle behind imaging with a pinhole camera and serves as a good approximation for lens-based imaging for the applications considered here. In the pinhole camera model, rays (or photons) from the scene are projected onto a planar screen after passing through a pinhole. The screen is typically called the image plane of the camera. Consider a camera with its pinhole at the origin and the image plane aligned with the plane $z = f$. Under this setup, a three-dimensional (3D) point $\mathbf{x} = (x, y, z)^T$ projects onto the image plane point $\mathbf{u} = (u, v)^T$ such that

$$u = f\frac{x}{z}, \quad v = f\frac{y}{z}. \tag{1}$$

We introduce the notation of *homogeneous coordinates*. We use the *tilde* ($\widetilde{\cdot}$) notation to represent entities in homogeneous coordinates. Given a $d-$dimensional vector $\mathbf{u} \in \mathbb{R}^{\mathbf{d}}$, its homogeneous representation is given as a $(d+1)-$dimensional vector $\widetilde{\mathbf{u}} \backsim [\mathbf{u}, 1]^T$, where the operator $\backsim$ denotes equality up to scale, i.e., $\widetilde{\mathbf{u}} \backsim \widetilde{\mathbf{x}} \leftrightarrow \widetilde{\mathbf{u}} = \lambda\widetilde{\mathbf{x}}, \lambda \neq 0$. Dealing with homogeneous quantities allows for a scale ambiguity in our representation that enables elegant representations of the basic imaging equation.

The pinhole camera maps a 3D world onto a 2D plane, and hence, the mapping (1) is many-to-one and non-invertible. All points that lie on a line passing through the pinhole map onto the same image plane point. Given a point on the image plane $\mathbf{u}$, its *pre-image* is defined as the set of all scene points that map onto $\mathbf{u}$ under central projection. It is easily seen that the pre-image of a point is a line in the real world. Without additional knowledge of the scene and/or additional constraints, it is not possible to identify the scene point that projects onto $\mathbf{u}$. This lack of invertibility leads to some of the classical problems in computer vision, the most fundamental being establishing correspondences across views.

There is one special scenario where the imaging equation becomes invertible on, and that is when the world is planar (see Figure 1). Most urban scenarios are a good fit for the model, as the majority of the actions in the world occur over the ground plane. This makes it a valid assumption for a host of visual sensing applications. The invertibility can also be efficiently exploited by algorithms for various purposes. Consider two views of a planar scene labeled View A and View B. Let $\mathbf{u}_A$ and $\mathbf{u}_B$ be the projections of a 3D point $\mathbf{x}$ lying on the plane onto views A and B respectively. The homography constraint introduces the following relationship between $\mathbf{u}_A$ and $\mathbf{u}_B$.

$$\widetilde{\mathbf{u}}_B \quad \backsim \quad H\widetilde{\mathbf{u}}_A \tag{2}$$

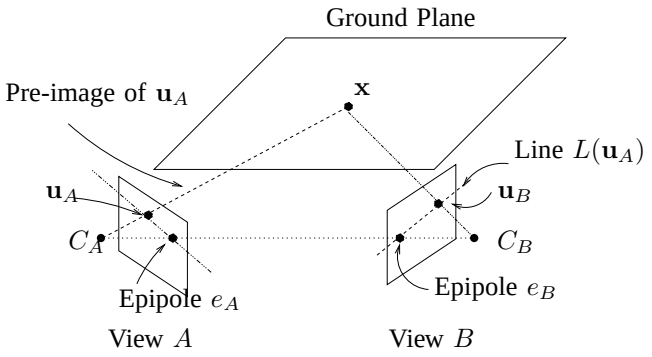

Fig. 1. Consider Views A and B (camera centers C_A and C_B) of a scene with a point $\mathbf{x}$ imaged as $\mathbf{u}_A$ and $\mathbf{u}_B$ on the two views. Without any additional assumptions, given $\mathbf{u}_A$, we can only constrain $\mathbf{u}_B$ to lie along the image of the pre-image of $\mathbf{u}_A$ (a line). However, if the world were planar (and we knew the relevant calibration information), then we could uniquely invert $\mathbf{u}_A$ to obtain $\mathbf{x}$ and re-project $\mathbf{x}$ to obtain $\mathbf{u}_B$

where H is a 3×3 invertible matrix. This implies that a point $\mathbf{u}_A$ in View A maps to the point $\mathbf{u}_B$ in View B as defined by the relationship in (2). The 3×3 matrix H (in (2)) is called the homography matrix or merely the homography. Note that H is a homogeneous matrix, and the transformation defined by it is unchanged when H is scaled. Further, H is invertible when the world plane does not pass through pinholes at either of the two views.

The premise of the induced homography critically depends on the fact that the pre-image of a point on the image plane is a unique point on the world plane. If we use a local 2D coordinate system over the world plane, then the image plane to world plane transformation (from their respectively 2D coordinate systems) can be shown to be a projective transformation, which as before can be encoded as a 3×3 homogeneous matrix, say H_π. This transformation is useful when we wish to estimate metric quantities, or quantities in an Euclidean setting. The most common example of this is when we need to localize the target in the scene coordinates.

B. Detection

The first and foremost task in distributed visual sensing is to detect objects of interest as they appear in the individual camera views [2], [3], [4]. In typical visual sensing scenarios, the objects of interest are those that are moving. Detection of moving objects is a much easier task, because object motion typically leads to changes in the observed intensity at the corresponding pixel locations, which can be used to detect moving objects.

Detection of moving objects is typically performed by modeling the static background and looking for regions in the image that violate this model. The simplest model is that of a single template image representing the static background. A test image can then be subtracted from the template and pixels with large absolute difference can be marked as *moving*. This simple model introduces the idea of background subtraction — essentially the process of removing static background pixels from an image.

There exist a host of background subtraction algorithms that work very well in many scenarios [2], [3], [4]. We denote the image frame at time t as I_t and background model as B_t, and the value/model of their $i-$th pixel as I_t^i and B_t^i. A commonly used model represents each pixel as Gaussian with mean μ_t^i and variance $\sigma_{i,t}^2$. The

background model is defined as:

$$\Pr(I_t^i|B_t) \propto \exp\left(-\frac{(I_t^i - \mu_t^i)^2}{2\sigma_{i,t}^2}\right). \quad (3)$$

Another background model that robustly handles periodic background disturbances is the mixture of Gaussians (MoG) model [4]. This model adaptively learns a MoG distribution at each pixel. The multi-modality of the underlying distribution gives the ability to capture repetitive actions as part of the background. An adaptive learning method can be used to keep track of global changes in the scene. As before, given a new image, we can compute the likelihood image and threshold it to obtain pixels that violate the background model. Figure 2 demonstrates results of background subtraction on multiple views of a scene and projects the representative points from subtracted results in each view to a top-down view of the ground plane. Spatial proximity on the ground plane can now be used to cluster points from different views and obtain a multi-view estimate of location of targets on the plane. We describe a systematic method to model and solve this fusion problem next.

C. Multi-view fusion and tracking

Once the objects of interest have been detected in each of the individual cameras, the next task is to track each object using multi-view inputs. Most algorithms maintain an appearance model for the detected objects and use this appearance model in conjunction with a motion model for the object to estimate the object position at each individual camera.

There exist many application domains that benefit immensely from multi-view inputs. Presence of multi-view inputs allows for the robust estimation of pose and limb geometry (markerless motion capture) [6], [7], [8]. When targets are at lower resolution, position tracking in scene coordinates provides information that is useful for higher level reasoning of the scene. As an example, [9] uses multiple cameras to track football players on a ground plane. Similarly, [10], [11], [12], [13], [14] consider the problem of multi-view tracking in the context of a ground plane, with the intent of localization of target position on the plane.

In the case of multi-camera systems, another important challenge that presents itself is the association of objects across the camera views. For cameras, that have non-overlapping fields of views, object association can be achieved by learning the relationship between patterns of entry and exit in various camera views [15]. For cameras with overlapping fields of views, an important issue that arises is fusion of object location estimates. This requires the use of epipolar geometry and triangulation in the most general case. As a special case, in the presence of ground-plane constraint, it is possible to derive efficient estimators for fusing multi-view information. We discuss this next.

Multi-camera tracking in the presence of ground-plane constraint has been the focus of many recent papers [16], [17], [13], [14]. The key concepts in many of the proposed algorithms are:

- **Association of data across views by exploiting the homography constraint:** This can be done by projecting various features associated with the silhouette. The vertical axis from each segmented human is used as the feature in [13], while points features [14], [16] or even the whole silhouette [17] form alternate choices. These

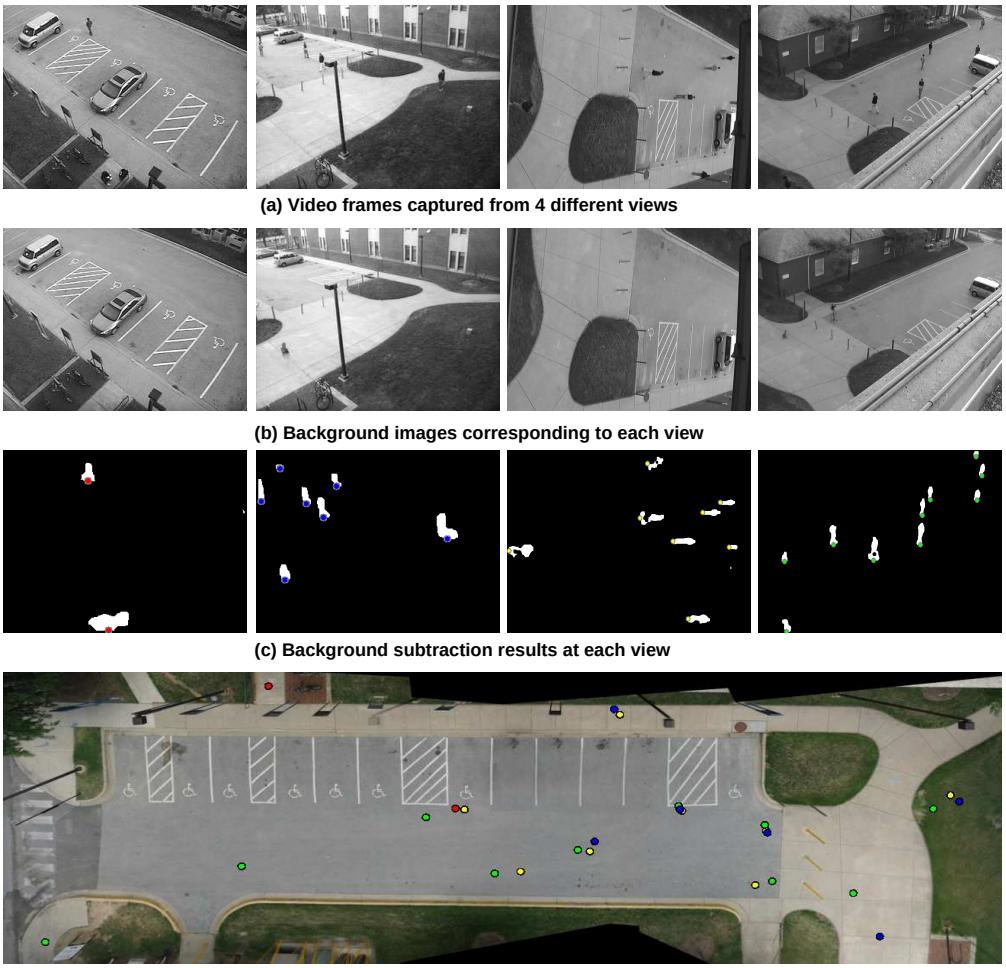

Fig. 2. Use of geometry in multi-view detection. (a) Snapshot from each camera. (b) Object free background image. (c) Background subtraction results. (d) Synthetically generated top view of the ground plane. The bottom point (feet) of each blob is mapped to the ground-plane using the image-plane to ground-plane homography. Each color represents a blob detected in a different camera view. Points of different colors very close together on the ground plane most likely correspond to the same subject seen via different camera views. Figure courtesy of [5].

feature(s) are projected onto a reference view using the homography transformation and consensus between features is used to associate data across views.

- **Temporal continuity of object motion to track:** Typically, a particle filter [13], [14] is used to temporally filter the data after association. Alternatively, [17] builds a temporal consistency graph and uses graph-cuts [18] to segment tracks.

Since the early work of Smith and Cheeseman [19], researchers have known that one has to account for the effect of varying covariances on parameter fusion and estimation accuracy, especially under highly adhoc placement of cameras. Consider, for example, the problem of location estimation of a point object moving on a plane. At each camera, background subtraction provides an estimate of where the object lies. We can now project the image

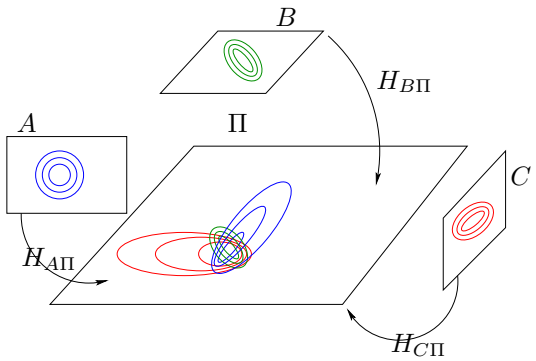

Fig. 3. A schematic showing densities in the image planes of cameras and their transformations to the ground plane.

plane locations to arrive at individual estimates of the world plane location of the tracked point. In an ideal noise-free condition, the estimates arising from each of the cameras would be identical. However, in the presence of noise corrupting the image plane observations, errors in calibration and inaccuracies in modeling, the world plane location estimates will no longer be identical. We now need a strategy to fuse these estimates. However, to do so in a systematic fashion we need to characterize the statistical properties of these estimates. Let us suppose that we have a characterization of the location of the object in the individual image planes as a random variable. We can project these random variables to the ground plane using the projective transformation linking the individual image planes and the ground plane.

Figure 3 shows an example of three cameras A, B, and C looking at a plane Π, with the image plane of B parallel to Π. In contrast, the image planes of A and C are perpendicular to Π. Also shown on the image planes of the cameras are uncertainty contours representing the image plane distribution at each camera. The homographies between the cameras and the plane Π are $H_{A\Pi}$, $H_{B\Pi}$ and $H_{C\Pi}$ respectively. In this setup, $H_{B\Pi}$ is not fully projective, defining only an affine transformation, as opposed to $H_{A\Pi}$ and $H_{C\Pi}$ which induce strong projective distortion. We would expect the density on B to retain its original form (similar error iso-contours) when projected on the plane.

The projective mapping is in general a non-linear transformation involving ratios. The statistics of random variables, when transformed under such a ratio transformation, change significantly. Given that the projective transformations linking different views of the same scene are different, one can expect that the statistics of random variables on the world plane arising from different views will necessarily be different, even when the original random variables are identically distributed.

Given M cameras and the homography matrices $H_i, i = 1, \ldots, M$ between the camera views and the ground plane, one can derive an algorithm for fusing location estimates. Let $\mathbf{Z}_u^i$ be the random variable modeling the object location on the image plane of the i-th camera. Let us assume that the random variables $\{\mathbf{Z}_u^i\}_{i=1}^M$ are statistically independent. Now, each of these random variables can be projected to the world plane to obtain $\mathbf{Z}_x^i$, such that $\widetilde{\mathbf{Z}}_x^i \backsim H_i \widetilde{\mathbf{Z}}_U^i, i = 1, \ldots, M$.

Fig. 4. Variance Ellipses are shown for the individual image planes. The corresponding color coded ellipse on the ground plane shows the covariance when transformed to the ground plane. The ellipse in black (on the ground plane) depicts the variance of the minimum variance estimator. Note that this estimate performs better than the individual estimates. Figure courtesy of [5].

Consider the distribution of $\mathbf{Z}_x^i$ under the assumption that the $\mathbf{Z}_u^i$ are jointly Gaussian. Specifically, when certain geometric properties are satisfied,[1] we can show that the distribution of Z_x^i is closely approximated by a Gaussian distribution [20], [14]. Further, we can relate the mean and the covariance matrix of the transformed random variable to the statistics of the original random variable and the parameters characterizing the projective transformation. This result is useful for designing strategies to fuse $\{\mathbf{Z}_x^i, i = 1, \ldots, M\}$ in an optimal sense.

In the case of multi-view localization, if the covariances of the estimates $\mathbf{Z}_x^i$ is Σ_i, then the minimum variance estimate $\mathbf{Z}_{mv}$ is computed as

$$\mathbf{Z}_{mv} = \sum_{i=1}^{M} \Sigma_i^{-1} \left(\sum_{j=1}^{M} \Sigma_j^{-1} \right)^{-1} \mathbf{Z}_x^i. \tag{4}$$

The covariance of the $\mathbf{Z}_{mv}$, Σ_{mv} is given as

$$\Sigma_{mv} = \left(\sum_{j=1}^{M} \Sigma_j^{-1} \right)^{-1}. \tag{5}$$

We refer the reader to [21], [19], [22], [14] for details of the derivation.

Hence, given a true object location on the ground plane, Σ_{mv} provides an estimate of the maximum accuracy (or minimum error) with which we can localize the object on the ground plane given modeling assumptions on the image plane (see Figure 4).

Finally, we can embed the concept used in constructing the minimum variance estimators to formulate a dynamical system that can be used to track objects using multi-view inputs. As before, we efficiently fuse estimates arising from different views by appropriately determining the accuracy of the estimates characterized by their covariance

[1]The required geometric properties reduce to the region of interest that is being imaged to be far away from the line of infinity in each of the views (for details, see [14]).

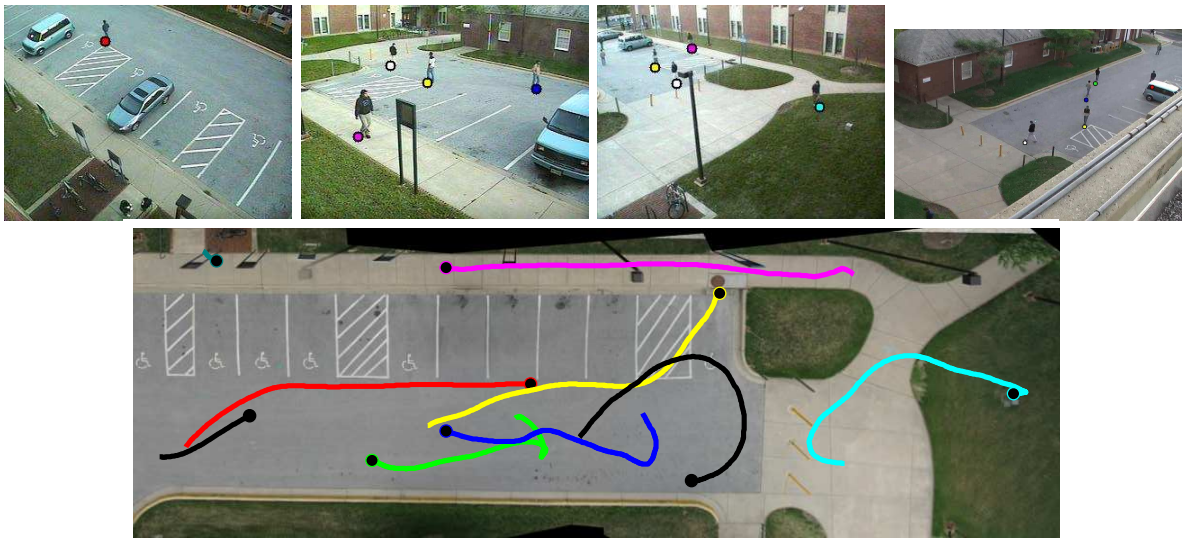

Fig. 5. Output from the multi-object tracking algorithm working with input from six camera views. (Top Row) Four camera views of a scene with several humans walking. Each camera independently detects/tracks the humans using a simple background subtraction scheme. The center location of the feet of each human is indicated with color coded circles in each view. These estimates are then fused taking into account the relationship between each view and the ground-plane. (Bottom row) Fused tracks overlaid on a top-down view of the ground-plane. Figure courtesy of [5].

matrices. Figure 5 shows tracking outputs from processing video data acquired from six cameras. Each object is tracked using a particle filter, and object to data associations are maintained using Joint Probability Data Association (JPDA) [23].

The JPDA algorithm can be easily implemented in a distributed sensor network. Each camera transmits the blobs extracted from the background subtraction algorithm to other nodes in the network. For the purposes of tracking, it is adequate even if we approximate the blob with an enclosing bounding box. Each camera maintains a multi-object tracker filtering the outputs received from all the other nodes (along with its own output). Further, the data association problem between the tracker and the data is solved at each node separately and the association with maximum likelihood is transmitted along with data to other nodes. Finally, the tracking problem can be robustly modeled as a non-linear dynamical system with the location and velocity of the targets as the state space of the system, the projected random variables Z_x^i forming a partial observation of the state with non-linearities introduced for robustness [14]. Here, the tracking problem is reformulated as one of Bayesian inference for the dynamical system. While exact inference is analytically intractable, approximate Bayesian inference for this dynamical system can be performed using the particle filter [24]. Toward this end, efficient implementations of particle filters influence multiple problems. In addition to faster run-time, these implementations help in reducing the resource utilization of the camera network as well. We discuss this next.

III. Efficient particle filtering

Particle filtering has been applied to a wide variety of problems such as tracking, navigation, detection and video-based object recognition. This generality of particle filters comes from a sample (or particle) based approximation of the posterior density of the state vector. This allows the filter to handle both the non-linearity of the system as well as the non-Gaussian nature of noise processes. However, the resulting algorithm is computationally intensive and as a result, the need for efficient implementations of the algorithm, tuned specifically towards hardware or multi-processor-based implementations.

Particle filtering involves three main modules: proposition, weight evaluation and resampling modules. Standard implementations of particle filtering typically use what is commonly known as *systematic resampling* (SR). Systematic resampling poses a significant challenge for pipelined implementations as it can only begin when all the weights are computed at the weight computation stage, and the cumulative sum of the weights is available. This means that any pipelined implementation would start the resampling only after all the weights are computed. This increases the latency of the whole implementation.

We first briefly describe particle filtering algorithms, and propose modifications to the general filtering framework that make it highly amenable to pipelined and parallel implementations.

A. Particle filtering: A brief overview

Particle filtering addresses the problem of Bayesian inference for dynamical systems. Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^p$ denote the state space and the observation space of the system respectively. Let $x_t \in \mathcal{X}$ denote the state at time t, and $y_t \in \mathcal{Y}$ the noisy observation at time t. We model the state sequence $\{x_t\}$ as a Markovian random process. Further we assume that the observations $\{y_t\}$ to be conditionally independent given the state sequence. Under these assumptions, the system is completely characterized by the following:

- $p(x_t|x_{t-1})$: The state transition probability density function, describing the evolution of the system from time $t-1$ to t. Alternatively, the same could be described with a *state transition model* of the form $x_t = h(x_{t-1}, n_t)$, where n_t is a noise process.
- $p(y_t|x_t)$: Observation likelihood density, describing the conditional likelihood of observation given state. As before, this relationship could be in the form of an *observation model* $y_t = f(x_t, \omega_t)$ where ω_t is a noise process independent of n_t.
- $p(x_0)$: The prior state probability at $t = 0$.

Given statistical descriptions of the models and noisy observations, we are interested in making inferences about the state of the system at current time. Specifically, given the observations till time $t, y_{1:t} = \{y_1, \ldots, y_t\}$, we would like to estimate the posterior density function $\pi_t = p(x_t|y_{1:t})$. With the posterior, we aim to make inferences $I(f_t)$ of the form

$$I(f_t) = \mathbf{E}_{\pi_t}[f_t(x_t)] = \int f_t(x_t) p(x_t|y_{1:t}) dx_t, \tag{6}$$

where f_t is some function of interest. An example of such an inference could be the conditional mean, where $f_t(x_t) = x_t$.

Under Markovian assumption on the state space dynamics and conditional independence assumption on the observation model, the posterior probability is recursively estimated using the Bayes theorem

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)\int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}}{p(y_t|y_{1:t-1})}. \tag{7}$$

Particle filtering approximates the posterior π_t with a discrete set of particles or samples $\{x_t^{(i)}\}_{i=1}^N$ with associated weights $\{w_t^{(i)}\}_{i=1}^N$ suitably normalized so that $\sum_{i=1}^N w_t^{(i)} = 1$. The approximation for the posterior density is given by

$$\hat{\pi}_t(x_t) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_t}\left(x_t^{(i)}\right) \tag{8}$$

where $\delta_{x_t}(\cdot)$ is the Dirac Delta function centered at x_t. The set $S_t = \{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ is the weighted particle set that represents the posterior density at time t, and is estimated recursively from S_{t-1}. The initial particle set S_0 is obtained from sampling the prior density $\pi_0 = p(x_0)$.

Particle filters sequentially generate S_t from S_{t-1} using the following steps:

1) **Importance Sampling:** Sample $x_t^{(i)} \sim g(x_t|x_{t-1}^{(i)}y_t), i = 1, \ldots, N$. This step is also called the *proposal step* and $g(\cdot)$ is called the *proposal density*.
2) **Computing Importance Weights:** Compute the unnormalized importance weights $\tilde{w}_t^{(i)}$,

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)})}{g(x_t^{(i)}|x_{t-1}^{(i)}y_t)},\ i = 1, \ldots, N. \tag{9}$$

3) **Normalize Weights:** Obtain the normalized weights $w_t^{(i)}$,

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}},\ i = 1, \ldots, N. \tag{10}$$

4) **Inference Estimation:** An estimate of the inference $I(f_t)$ is given by

$$\hat{I}_N(f_t) = \sum_{i=1}^{N} f_t(x_t^{(i)}) w_t^{(i)} \tag{11}$$

5) **Resampling:** To avoid sample degeneracies, the weighted sample set $\{x_t^{(i)}, w_t^{(i)}\}$ is resampled to get an unweighted sample set $S_t = \{\tilde{x}_t^{(i)}\}$.

The resampling step, at the end of each iteration is performed to avoid certain degeneracies in the finite sample representation of the posterior density. After a few time steps, all importance weights, except a few, converge to zero. These weights will remain at zero for all future time instants (as a result of (9)), and do not contribute to the estimation of $\hat{I}_N(f_t)$. This is circumvented by the introduction of a *resampling* step. Resampling essentially replicates particles with higher weights and eliminates those with low weights. The most popular resampling scheme, called *systematic resampling* (SR) [24], samples N particles from the set $\{x_t^{(i)}\}$ (samples generated after proposal) according to the multi-nomial distribution with parameters $w_t^{(i)}$ to get a new set of N particles S_t. The next iteration uses this new set $\tilde{S}_t$ for sequential estimation.

Particle filtering algorithms that use Sequential Importance Sampling (SIS) and SR are collectively called SISR algorithms. Systematic resampling is a computationally tricky as it requires the knowledge of the normalized weights. Resampling based on SR cannot start until all the particles are generated and the value of the cumulative sum is known. This is the basic limitation that we overcome by proposing alternative techniques.

B. Metropolis Hastings algorithm

Particle filtering is a special case of more general MCMC based density sampling techniques that are especially tuned for dynamical systems. The Metropolis Hastings Algorithm (MHA) [25], [26] is considered the most general MCMC sampler. Popular samplers such as the Metropolis Sampler [27] or the Gibbs Sampler [28] are special cases of this algorithm. We first present the general theory of MCMC sampling using the MHA algorithm and then state the conditions under which the general theory fits into the particle filtering algorithm presented before.

The MHA generates samples from the desired density (say $p(x), x \in \mathcal{X}$) by generating samples from an easy to sample *proposal distribution*, say $q(x|y), x \in \mathcal{X}, y \in \mathcal{X}$. MHA produces a sequence of states $\{x^{(n)}, n \geq 0\}$, which by construction is Markovian in nature, through the following iterations.

1) Initialize the chain with an arbitrary value $x^{(0)} = x_0$. Here, x_0 could be user specified.
2) Given $x^{(n)}, n \geq 0$, generate $\hat{x} \backsim g(\cdot|x^{(n)})$, where g is the sampling or proposal function.
3) Accept $\hat{x}$ with probability

$$\alpha(x^{(n)}, \hat{x}) = \min\left\{\frac{p(\hat{x})}{p(x^{(n)})}\frac{g(x^{(n)}|\hat{x})}{g(\hat{x}|x^{(n)})}, 1\right\} \tag{12}$$

That is, for a uniform random variable $u \backsim U[0,1]$

$$x^{(n+1)} = \begin{cases} \hat{x} & \text{if} \quad u \leq \alpha(x^{(n)}, \hat{x}) \\ x^{(n)} & \text{otherwise.} \end{cases} \tag{13}$$

A special case of the MHA is the Independent Metropolis Hastings Algorithm (IMHA) where the proposal function $g(x|y)$ is set as $g(x)$. This makes the proposal function independent of the previously accepted sample in the chain. This would mean that the acceptance probability (12) $\alpha(x^{(n)}, \widehat{x})$ of a proposal $\widehat{x} \in \mathcal{X}$ with the chain at $x^{(n)} \in \mathcal{X}$,

$$\alpha(x^{(n)}, \widehat{x}) = \min\left\{\frac{p(\widehat{x})}{g(\widehat{x})}\frac{g(x^{(n)})}{p(x^{(n)})}, 1\right\} \tag{14}$$

Both IMHA and SISR are algorithms designed to generate samples according to a probability density function, with the SISR suited specifically to the sequential nature of dynamical systems. In this regard, the key difference between the IMHA and the SISR algorithm lies in the fact that the SISR algorithm requires the knowledge of cumulative sum of weights (the term $\sum_{j=1}^{N} \tilde{w}_t^{(j)}$ in (10)). This is important as the cumulative sum can only be computed when the weights corresponding to the whole particle set is known. Hence, SR can only begin after all particles have been generated and their weights have been computed. In contrast, the IMHA imposes no such bottlenecks. We show how this property can be exploited to design a filter that does not suffer from the bottle-necks introduced by SR.

C. Particle Filtering with IMHA-based resampling

The bottlenecks introduced by the SR technique can be overcome by using IMHA for resampling. However, there are some basic issues that need to be resolved before we achieve this. To begin with, the generation of particles using importance sampling works differently for the two algorithms. Particle filtering allows for the importance function to be defined locally for each particle. Mathematically, the i^{th} particle at time t is generated from an importance function, represented as $g(x_t|x_{t-1}^{(i)}y_t)$, parametrized by $x_{t-1}^{(i)}$. This poses a problem in the application of IMHA to estimate the posterior density, because the concept of importance functions associated with each particle does not extend to IMHA. In contrast, the MHA algorithm requires the importance function to depend functionally only on the last accepted sample in the chain, and in the case of the IMHA, the importance function remains the same.

Given a set of unweighted samples $\{x_{t-1}^{(i)}, i = 1, \ldots\}$ sampled from the posterior density $p(x_{t-1}|y_{1:t-1})$ at time $t-1$, we can approximate the posterior by

$$p(x_{t-1}|y_{1:t-1}) \approx \frac{1}{N}\sum_{i=1}^{N} \delta_{x_{t-1}}(x_{t-1}^{(i)}). \tag{15}$$

Using (7) and (15), we can approximate the posterior at time t as

$$p(x_t|y_{1:t}) \approx \frac{p(y_t|x_t)}{p(y_t|y_{1:t-1})}\frac{1}{N}\sum_{i=1}^{N} p(x_t|x_{t-1}^{(i)}). \tag{16}$$

Sampling from this density can be performed using MHA or IMHA. The issue of choice of importance function now arises. The importance function typically reflects and exploits the knowledge of application domain or could be a clever approximation to the posterior. For this reason, we would like to reuse the importance function corresponding to the underlying model.

Keeping this in mind, we propose a new importance function of the form,

$$g'(x_t|y_t) = \sum_{i=1}^{N} \frac{1}{N} g(x_t|x_{t-1}^{i}y_t) \tag{17}$$

Note that $g'(x_t|y_{1:t})$ qualifies to be an importance function for use in IMHA given the independence of the function on current state of the chain. To sample from $g'(x_t|y_t)$, we need to first sample $I \backsim U[1, 2, \ldots, N]$, and then sample from $g(\cdot|x_{t-1}^{I}y_t)$. The sampling of I can be done deterministically given the ease of sampling from uniform densities over finite discrete spaces. Finally, although the new importance function is functionally different from the one used in the SISR algorithm, the generated particles will be identical.

The overall algorithm proceeds similar to IMHA. We first propose particles using the new importance function $g'(x_t|y_{1:t})$. The acceptance probability now takes the form

$$\alpha(x_t, \widehat{x}) = \min\left\{\frac{w'(\widehat{x})}{w'(x_t)}, 1\right\} \tag{18}$$

with

$$w'(x_t) = p(y_t|x_t)\frac{\sum_{i=1}^{N} p(x_t|x_{t-1}^{(i)})}{\sum_{i=1}^{N} g(x_t|x_{t-1}^{(i)}y_t)} \tag{19}$$

Further, if the choice of the importance function were the same as the state transition model, i.e, $g(x_t|x_{t-1}y_t) = p(x_t|x_{t-1})$, then the acceptance probability becomes a ratio of likelihoods,

$$\alpha(x_t^{(n)}, \widehat{x}) = \min\left\{\frac{p(y_t|\widehat{x})}{p(y_t|x_t^{(n)})}, 1\right\} \tag{20}$$

We can now avoid the systematic resampling of traditional particle filtering algorithms and use IMHA to generate the unweighted particle set/stream from the desired posterior.

As before, we have an unweighted particle set S_{t-1}, that contains particles approximating the posterior at time $t-1$, $\pi_{t-1}(x_{t-1})$. We aim to estimate an approximation to the posterior at time t. As before, the algorithm is initialized with S_0 containing samples from the prior $p(x_0)$.

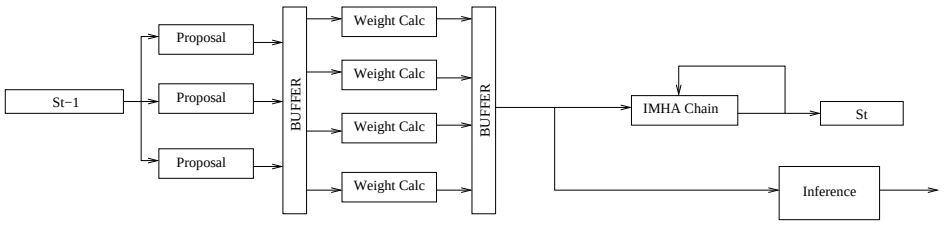

Fig. 6. Parallel Implementation of the IMHA-based particle filter with a single Metropolis-Hastings Chain.

The main steps are stated below (see Figure 6 for a block diagram of the implementation):

- **Importance Sampling (Step 1):** Generate $N + N_b$ indices $J(i), i = 1, \ldots, N + N_b$ uniformly from the set $\{1, 2, 3, \ldots, N\}$, where N_b is an estimate of the *burn in* period and N is the number of particles required. between $1..N$ with uniform density.
- **Importance Sampling (Step 2):** From the particle set $S_{t-1} = \{x_{t-1}^{(i)}, i = 1, \ldots, N\}$ at time $t-1$, propose $N + N_b$ particles to form the set $\widehat{S}_t = \{\widehat{x}_t^{(i)}, i = 1, \ldots, N + N_b\}$ using the rule:

$$\widehat{x}_t^{(i)} \backsim g(\cdot|x_{t-1}^{J(i)}y_t). \tag{21}$$

- **Compute Importance Weights:** For each particle in $\widehat{S}_t$, evaluate the importance weights $w'^{(i)}_t$, for each i using (19).
- **Inference:** Estimate the expected value of functions of interest. Compute

$$\widehat{I}_t(f_t) = \frac{\sum_{i=1}^{N+N_b} f(x_t^{(i)}) w'^{(i)}_t}{\sum_{i=1}^{N+N_b} w'^{(i)}_t}. \tag{22}$$

  Note that samples discarded during burn-in can still be used in the computation of (22) as the unnormalized particle set $\{x_t^{(i)}, w'^{(i)}_t, i = 1, \ldots, N + N_b\}$ *is still properly weighted (when normalized) [29].*
- **MCMC Sampler:** Use the IMH sampler to parse through the set $\widehat{S}_t$, to generate a new unweighted set of particles using the following steps.
  1) Initialize the chain with $x_t^{(1)} = \widehat{x}_t^{(1)}$ the first particle proposed.

2) for $i = 2, \ldots, N + N_b$,

$$x_t^{(i)} = \begin{cases} \widehat{x}_t^{(i)}, & \text{with prob.} \quad \alpha(x_t^{(i-1)}, \widehat{x}_t^{(i)}) \\ x_t^{(i-1)}, & \text{with prob.} \quad 1 - \alpha(x_t^{(i-1)}, \widehat{x}_t^{(i)}) \end{cases} \tag{23}$$

where $\alpha(\cdot, \cdot)$ is the acceptance probability as defined in (12).

Discarding the first N_b samples for burn in, the remaining N samples form $S_t = \{x_t^{(i)}, i = N_b + 1, \ldots, N\}$, the approximation of $p(x_t|y_{1:t})$.

We can now compare the algorithm given above with the classical SISR discussed in Section III-A. Note that the SISR algorithm involves a weight normalization step (equation (10)). However, the proposed algorithm works with ratios of unnormalized weights and requires no such normalization. This provides for the following advantages in the proposed methodology:

- The IMH sampler works with ratios of importance weights. This obviates the need for knowledge of normalized importance weights, as we can work with unnormalized weights. This allows the IMH sampler to start parsing through the particles as they are generated, and not wait for the entire particle set to be generated and the importance weights computed.
- In contrast, in SISR, the resampling can begin only when all particles are generated and the cumulative sum or normalized weights are known.

D. Experimental results

The design methodologies proposed have been verified for two applications: a synthetic example originally discussed in [24] and the problem of visual tracking . The computational testbed was the UMIACS Red/Blue cluster. The Red cluster consists of 16 PII (400 MHz) PCs running Redhat 7.3, with each PC having a RAM of 1GB. The Blue cluster consists of 12 PIII (550Mhz). We used MPICH [30][31], an implementation of the Message Passing Interface (MPI) for communication between threads.

We chose to implement over a multi-processor cluster framework since the underlying theory applies both to hardware based design as well as to clusters. In general, MPI has large overheads; however, such overheads are common and identical to both SISR as well as the MCMC based schemes. The conclusions from our experimental observations still remain the same. We implemented the particle filter based online tracking algorithm presented in [32] using the Red Cluster.

For comparison purposes, we also implemented the traditional SISR with the same specifications as the proposed methodology.

The algorithm was used to process 20 frames of a video sequence, tracking a car. Figure 7 shows typical tracking results. The filter was run with 840 and 1680 particles, with the number of cluster nodes for weight computation R_w varying. $R_w = 1$ corresponds to the sequential implementation, and $R_w > 1$ corresponds to the parallel implementation with a single chain. Under the same setup, we tried an implementation of SISR, replacing the IMH Chain with a systematic resampler. The main difference between the algorithms is that the systematic resampler could begin only when all particles were processed and the normalized weights are known.

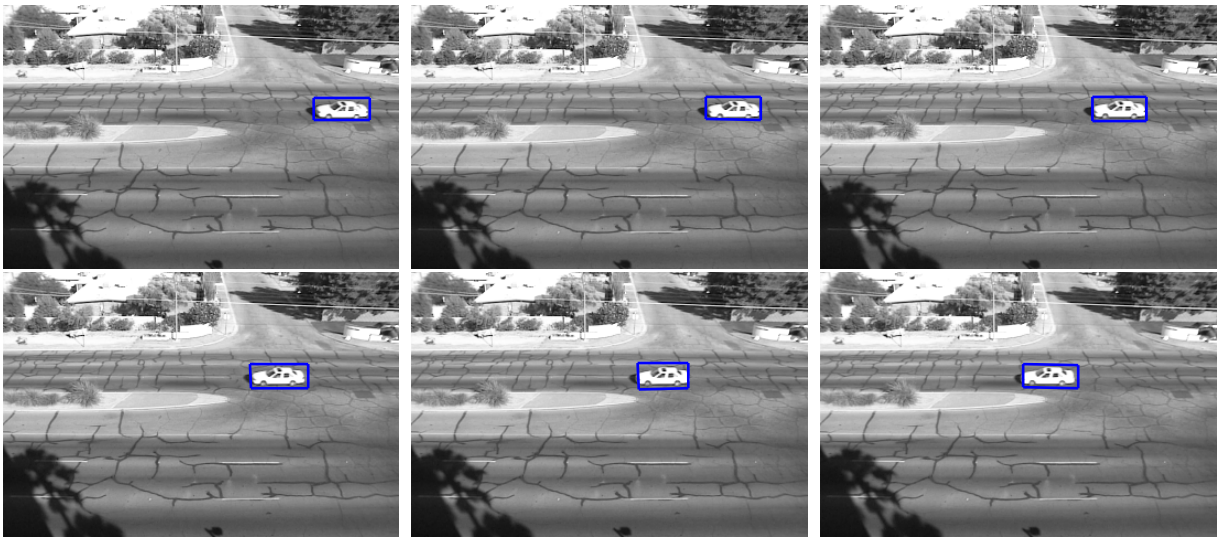

Fig. 7. Frames 1,4,8,.. of the tested set. The output of the tracker is inlaid on top. Figure courtesy of [33].

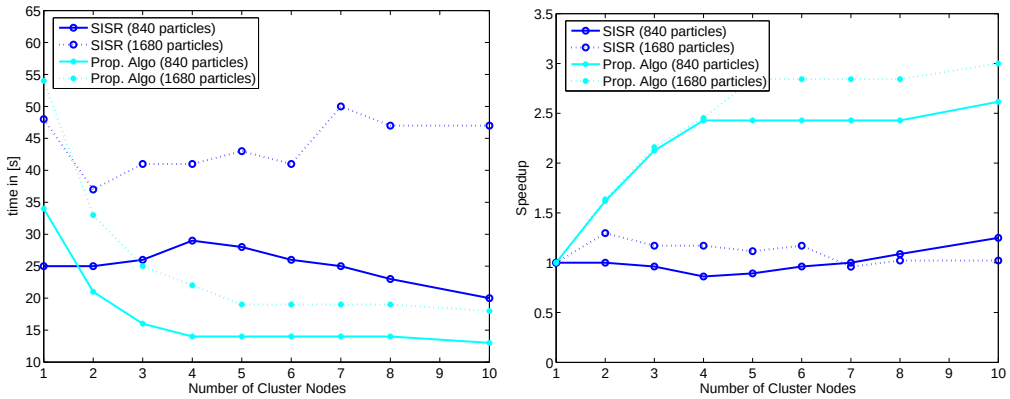

Fig. 8. (Left) Actual time (in seconds) taken to process 20 frames, with a filter of 840 particles, with varying number of R_w. (Right) Speedup obtained by replication of the weight computation node. Note the linear speedup obtained with the proposed algorithm. Figure courtesy of [33].

Figure 8 shows the actual time taken (in seconds) and the speedup to process 20 frames of video, with 840/1680 particles for the proposed algorithm and SISR. Note the $1/x$-like decay exhibited by the time taken by the proposed algorithm. The $1/x$-like behavior now translates to a linear increase in speedup with the number of processing nodes. The two plots demonstrate the pipelinability of the proposed algorithm. It can be seen that the speedup tapers-off as number of cluster nodes increases. This is attributed to increasing communication delays between the nodes. There are no standard models for communication delays when using MPICH. As we use more and more processors, inter-processor communication becomes the dominant source of delay, and further parallelization does not help.

The resampling method and the associated implementation schemes proposed here, allows for a pipeline that is free of bottle-necks. Further, implementations using the proposed methodologies show a speedups that increases

linearly with the number of processing nodes utilized. This allows us to parallelize the algorithm to achieve the desired runtime rate. In contrast, implementations based on SISR do not scale that easily with the number of the processing nodes used.

IV. Compressive sensing for background subtraction

The emerging field of compressive sensing (CS) provides an alternate way of handling the data deluge problem. CS relies on the fact that much of real world signals (such as images) are inherently sparse in some known linear basis such as wavelets. Sparsity of the signal hints at the fact that the inherent information content of a signal is significantly lower than its ambient dimension. This motivates the design of a parsimonious sensing strategy that samples the signal at its information rate. It has been shown that from under-sampled linear projections of the signal, the actual signal can be recovered by exploiting its sparsity. In addition to this, much of the subsequent processing that we are interested in can be performed directly on these compressive measurements, thereby obviating the need to reconstruct the high dimensional signal (image). Toward this end, in this section, we first introduce compressive sensing, and show that some of the fundamental operations in a multi-camera system such as background subtraction and multi-view tracking can be performed directly on these compressive measurements with a high level of fidelity. This leads to a sensor network that senses only at the information rate required for the task at hand.

A. Compressive sensing

Compressive sensing [34], [35] relates to the concept of reconstructing sparse signals from under-sampled linear measurements. Consider a signal $\mathbf{x} \in \mathbb{R}^N$, which is sparse in a basis $\mathbf{B}$, that is, $\mathbf{s} \in \mathbb{R}^N$ defined as $\mathbf{x} = \mathbf{B}\mathbf{s}$ is sparse. We call a vector K-sparse if it has atmost K non-zero components, or equivalently, if $\|\mathbf{s}\|_0 \leq K$, where $\|\cdot\|_0$ is the ℓ_0 norm or the number of non-zero components.

We are interested in the problem of sensing the signal $\mathbf{x}$ from linear measurements. Ideally, with no additional knowledge about $\mathbf{x}$, we would require N linear measurements of $\mathbf{x}$, which would then form an invertible linear system. The theory of compressed sensing shows that it is possible to reconstruct $\mathbf{x}$ from M measurements even when $M \ll N$ by exploiting the sparsity of $\mathbf{s} = \mathbf{B}^T\mathbf{x}$. Consider a measurement vector $\mathbf{y} \in \mathbb{R}^M$ obtained using a $M \times N$ measurement matrix $\mathbf{\Phi}$, such that

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} + e = \mathbf{\Phi}\mathbf{B}\mathbf{s} + e \tag{24}$$

where e is the measurement noise. For $M < N$, estimating $\mathbf{x}$ from the linear measurements is an ill-conditioned problem. However, when there exists a basis $\mathbf{B}$ such that $\mathbf{s}$ as defined above is K sparse, then compressive sensing allows for recovery of $\mathbf{s}$ (or alternatively, $\mathbf{x}$) from $M = O(K\log(N/K))$ measurements. In particular, when $\mathbf{B}$ is a fixed basis, it can be shown that using a randomly generated measurement matrix Φ allows for the recovery of $\mathbf{x}$ with a high probability. Typical choices for such measurement matrix are the Bernoulli matrices and the Gaussian matrix [34]. Such randomly generated matrices satisfy the *Restricted Isometry Property* [36], [37], which ensures that all sub-matrices formed by choosing (a certain number of) columns of $\mathbf{\Phi}$ are orthogonal (and hence, invertible) with a high probability.

Estimating K sparse vectors that satisfy the measurement equation of (24) can be formulated as the following ℓ_0 optimization problem:

$$(P0): \ \min \|\mathbf{s}\|_0 \text{ s.t. } \|\mathbf{y} - \mathbf{\Phi B x}\|_2 \leq \epsilon \tag{25}$$

where the ℓ_0 norm, $\|\cdot\|_0$ counts the number of non-zero elements. This is typically a NP-hard problem. However, the equivalence between ℓ_0 and ℓ_1 norm for such systems [38] allows us to reformulate the problem as one of ℓ_1 norm minimization.

$$(P1): \ \min \|\mathbf{s}\|_1 \text{ s.t. } \|\mathbf{y} - \mathbf{\Phi B s}\| \leq \epsilon \tag{26}$$

with ϵ being a bound for the measurement noise e in (24). It can be shown that the solution to the (P1) is with a high probability the K sparse solution that we seek. There exist a wide range of algorithms that solve $P1$ to various approximations or reformulations [39][40][41][35]. Most of them note that the problem $(P1)$ is a convex problem, and in particular, can be recast as a Second Order Cone Program (SOCP) for which there exist efficient numerical techniques.

B. Compressive background subtraction

For background subtraction, our objective is to recover the location, shape and appearance of the objects given a test image over a known background. Let us denote the background, test, and difference images as $\boldsymbol{x}_b$, $\boldsymbol{x}_t$, and $\boldsymbol{x}_d$, respectively. The difference image is obtained by pixel-wise subtraction of the background image from the test image. Note that the support of $\boldsymbol{x}_d$, denoted as $\mathcal{S}_d = \{n | n = 1, \ldots, N; |\boldsymbol{x}_d(n)| \neq 0\}$, gives us the location and the silhouettes of the objects of interest, but not their appearance.

Suppose that $\boldsymbol{x}_b$ and $\boldsymbol{x}_t$ are typical real-world images in the sense that when wavelets are used as the sparsity basis for $\boldsymbol{x}_b$, $\boldsymbol{x}_t$, and $\boldsymbol{x}_d$, these images can be well approximated with the largest K coefficients with hard thresholding [42]. The images $\boldsymbol{x}_b$ and $\boldsymbol{x}_t$ differ only on the support of the foreground, which has a cardinality of $P = |\mathcal{S}_d|$ pixels with $P \ll N$. As a consequence, we can expect to require a much smaller number of samples to reconstruct the difference image than the background or foreground images.

Let us assume that we have multiple compressive measurements $\boldsymbol{y}_{bi}$ ($M \times 1$, $i = 1, \ldots, B$) of training background images $\boldsymbol{x}_{bi}$, where $\boldsymbol{x}_b$ is their mean. Each compressive measurement is a random projection of the whole image, whose distribution we approximate as an i.i.d. Gaussian distribution with a constant variance $\boldsymbol{y}_{bi} \sim \mathcal{N}\left(\boldsymbol{y}_b, \sigma^2 \boldsymbol{I}\right)$, where the mean value is $\boldsymbol{y}_b = \mathbf{\Phi} \boldsymbol{x}_b$. When the scene changes to include an object which was not part of the background model and we take the compressive measurements, we obtain a test vector $\boldsymbol{y}_t = \mathbf{\Phi} \boldsymbol{x}_t$, where $\boldsymbol{x}_d = \boldsymbol{x}_t - \boldsymbol{x}_b$ is sparse in the spatial domain.

In general, the sizes of the foreground objects are relatively smaller than the size of the background image; hence, we model the distribution of the elements of the *literally* background subtracted vector as $\boldsymbol{y}_d = \boldsymbol{y}_t - \boldsymbol{y}_b \sim \mathcal{N}\left(\boldsymbol{\mu}_d, \sigma^2 \boldsymbol{I}\right)$ ($M \times 1$). Note that the appearance of the objects constructed from the samples $\boldsymbol{y}_d$ would correspond to the literal subtraction of the test frame and the background; however, their silhouette is preserved.

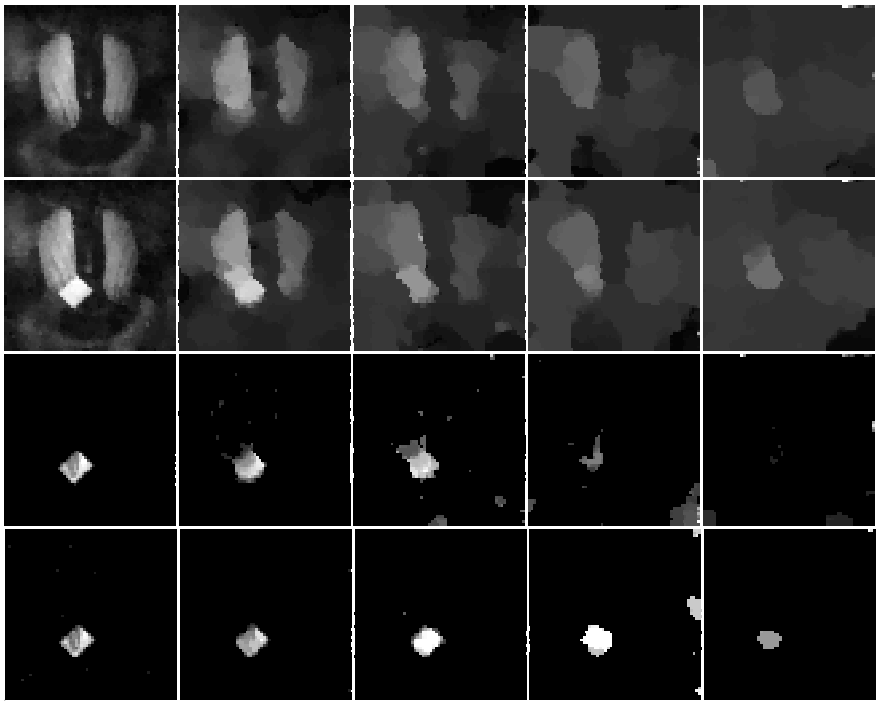

Fig. 9. Background subtraction experimental results using an SPC. Reconstruction of background image (top row) and test image (second row) from compressive measurements. Third row: conventional subtraction using the above images. Fourth row: reconstruction of difference image directly from compressive measurements. The columns correspond to measurement rates M/N of 50%, 5%, 2%, 1% and 0.5%, from left to right. Background subtraction from compressive measurements is feasible at lower measurement rates than standard background subtraction. Figure courtesy of [44].

We performed background subtraction experiments with a single pixel camera [43] ; in our test, the background $\boldsymbol{x}_b$ consists of the standard test *Mandrill* image, with the foreground $\boldsymbol{x}_t$ consisting of a white rectangular patch as shown in Fig. 9. Both the background and the foreground were acquired using pseudorandom compressive measurements ($\boldsymbol{y}_b$ and $\boldsymbol{y}_t$, respectively). We obtain measurements for the subtraction image as $\boldsymbol{y}_d = \boldsymbol{y}_t - \boldsymbol{y}_b$. We reconstructed both the background, test, and difference images, using total variation (TV) minimization. The reconstruction is performed using several measurement rates ranging from 0.5% to 50%. In each case, we compare the subtraction image reconstruction with the difference between the reconstructed test and background images. The resulting images are shown in Figure 9, and it can be seen that for low rates the background and test images are not recovered accurately, and therefore the subtraction performs poorly; however, the sparser foreground innovation is still recovered correctly from the difference of the measurements, with rates as low as 1% being able to recover the foreground at this low resolution.

C. Multi-view ground plane tracking

Background subtraction forms an important pre-processing component for many vision applications. In this regard, it is important to see if the imagery generated using compressive measurements can be used in such applications.

Fig. 10. Tracking results on a video sequence of 300 frames. Results are obtained using only $1/5$ the bandwidth required of traditional central servers. (Left) The first two rows show sample images and background subtraction results using the compressive measurements, respectively. The background subtracted blobs are used to detect target location on the ground plane. (Right) The detected points using CS (blue dots) as well as the detected points using full images (black). The distances are in meters. Figure courtesy of [44].

We demonstrate a multi-view tracking application where accurate background subtraction is key in determining overall system performance.

In Figure 10, we show results on a multi-view ground plane tracking algorithm over a sequence of 300 frames with 20% compression ratio. We first obtain the object silhouettes using the compressive samples at each view. We use wavelets as the sparsifying basis $\mathbf{B}$. At each time instant, the silhouettes are mapped on to the ground planes and averaged. Objects on the ground plane (e.g., the feet) combine in synergy while those off the plane are in parallax and do not support each other. We then threshold to obtain potential target locations as in [17]. The outputs indicate the background subtracted images are sufficient to generate detections that compare well against the detections generated using the full non-compressed images. Hence, using our method, the communication bandwidth of a multi-camera localization system can be reduced to one-fifth if the estimation is performed at a central location.

V. Conclusions and Future directions

In this chapter, we have discussed some of the main challenges in distributed video processing, namely robust and computationally efficient inference, and opportunistic and parsimonious sensing. Advances in these are crucial to the deployment of large camera networks. We conclude by highlighting several recent trends that we believe will have significant impact on various aspects of distributed video processing.

A. Distributed Bayesian Inference

The algorithms for tracking presented here were both based on online inference using particle filtering. Therefore, in order to make these algorithms truly distributed and enable their implementation on huge camera networks containing hundreds of cameras, one needs to pay attention to methods that enable these particle-filter-based estimates to be performed in a distributed manner. This can be achieved using either synchronized particle filtering or the more general means of distributed function estimation.

1) Synchronized Particle Filtering: One way to decentralize the filter operations is to replicate it identically at each node. For particle filtering, this can be done easily if the random number generators are made identical across

nodes. Such a scheme is referred to as synchronized particle filtering [45]. By initializing the random number generator with the same seed, all nodes can be made to generate the same particles, which in turn makes fusion of the associated weights simpler. The communication costs are then limited to the transmission of the associated weights across the whole network. The immense flexibility of this approach allows for it to be effective in any particle-filtering algorithm. However, this freedom in generality comes with associated drawbacks. For one, the stability of the algorithm depends critically on the requirement of synchronized random numbers, which requires that the hardware at each node be the same. Further, this particular way of decentralization does not efficiently use the processing power of the nodes, as in the end the same computations are performed identically at each node.

2) Distributed Function Estimation: We can relax the need to make our distributed inference algorithm identical to the centralized one. There are a host of methods that allow for the computation of average mean through explicit global communication or through local consensus [46]. An alternative to the concept of synchronous filtering can be by approximating the inference at each camera with a Gaussian mixture model [47] or, in general, any parametric density family. The parameters can then be transmitted to all nodes in the sensor network, each of which locally updates their densities. In addition to distributed inference and filtering, efficient implementations of particle filters form an important direction for future research, especially in the context of decentralized computing and sensing. Existing approaches to this problem [48] are limited to node-level algorithms, although under a distributed architecture of computing.

B. Manifold-based dimensionality reduction (NLDR)

Images and video data lead to extremely high-dimensional data sets and in order to represent, visualize, analyze, interpret, and process such high-dimensional data, one needs to encapsulate the salient characteristics of the data in a lower dimensional representation. Traditionally, this has been performed using linear dimensionality reduction techniques such as principal component analysis (PCA) [49] or independent component analysis (ICA) [50]. Such linear dimensionality reduction methods are limited in applicability for vision applications, because the geometric constraints imposed by the imaging device and the lighting characteristics lead to non-linear constraints on image and video data. Recent research in the fields of manifold learning and NLDR has led to an explosion of new results and algorithms designed to tackle such non-linear embeddings of the high-dimensional data.

NLDR approaches such as the locally linear embedding (LLE) [51], Hessian LLE [52], Isomap [53] , the Laplacian eigenmap [54] and local tangent space alignment (LTSA) [55] construct and represent the high-dimensional data using a low-dimensional representation. The parameters of the representation are obtained by optimizing an appropriate cost function that leads to embeddings that are Euclidean-like locally, but globally are highly nonlinear. Moreover, most of these techniques also have an elegant graph formulation that explicitly describes the local properties that these approaches preserve. Such approaches for NLDR find several natural and compelling applications in vision tasks, as both image and video data are inherently high-dimensional and lend themselves to non-linear dimensionality reduction.

Video sequences are also naturally amenable to analysis using nonlinear dimensionality reduction approaches.

As an example consider action analysis and activity recognition. The goal is to recognize the pose of the subject from images, then use the knowledge about pose to perform activity recognition. Since the human body can be modeled as a kinematic chain with small degrees of freedom and, in most cases, the camera is assumed static, this means that the obtained silhouettes of the subject span a low-dimensional space. If NLDR approaches were used to extract this low-dimensional representation, then problems such as pose recognition and activity recognition could be solved using these projections, as opposed to the original imaging data.

REFERENCES

[1] H. Aghajan and A. Cavallaro, *Multi-Camera Networks: Concepts and Applications*, Elsevier, 2008.

[2] T. Lv, B. Ozer, and W. Wolf, "A real-time background subtraction method with camera motion compensation," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2004, vol. 1, pp. 331–334.

[3] S. Joo and Q. Zheng, "A temporal variance-based moving target detector," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2005.

[4] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, 2002.

[5] A. C. Sankaranarayanan, A. Veeraraghavan, and R. Chellappa, "Object detection, tracking and recognition for multiple smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1606–1624, 2008.

[6] D. M. Gavrila and L. S. Davis, "3-D model-based tracking of humans in action: a multi-view approach," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 1996, pp. 73–80.

[7] A. Sundaresan and R. Chellappa, "Model driven segmentation of articulating humans in Laplacian Eigenspace," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1771–1785, 2008.

[8] Q. Delamarre and O. Faugeras, "3D articulated models and multi-view tracking with silhouettes," in *Proceedings of IEEE International Conference on Computer Vision*, 1999, vol. 2, pp. 716–721.

[9] M. Xu, J. Orwell, and G. Jones, "Tracking football players with multiple cameras," in *Proceedings of IEEE International Conference on Image Processing*, 2004, vol. 5, pp. 2909–2912.

[10] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking," in *Proceedings of Workshop on Motion and Video Computing*, 2002, pp. 169–174.

[11] A. Mittal and L. S. Davis, "M2 Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene," *International Journal of Computer Vision*, vol. 51, no. 3, pp. 189–203, 2003.

[12] S. M. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Proceedings of European Conference on Computer Vision*, 2006, vol. 4, pp. 133–146.

[13] K. Kim and L. S. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guded particle filtering," in *Proceedings of European Conference on Computer Vision*, 2006, vol. 3, pp. 98–109.

[14] A. C. Sankaranarayanan and R. Chellappa, "Optimal multi-view fusion of object locations," in *Proceedings of IEEE Workshop on Motion and Video Computing (WMVC)*, 2008, pp. 1–8.

[15] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society; 1999, 2004, vol. 2.

[16] F. Fleuret, J. Berclaz, and R. Lengagne, "Multi-camera people tracking with a probabilistic occupancy map," Tech. Rep., EPFL/CVLAB2006.07, July 2006.

[17] S. M. Khan and M. Shah, "A multi-view approach to tracking people in crowded scenes using a planar homography constraint," in *ECCV*, 2006, vol. 4, pp. 133–146.

[18] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[19] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*, Cambridge Univ. Press, 2003.

[21] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167–193, 1990.

[22] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*, Elsevier Science Inc. New York, NY, USA, 1996.

[23] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press Professional, Inc. San Diego, CA, USA, 1987.

[24] N. Gordon, D. Salmon, and A Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings on Radar and Signal Processing*, vol. 140, pp. 107–113, 1993.

[25] S. Chib and E. Greenberg, "Understanding the metropolis hastings algorithm," *American Statistician*, vol. 49, pp. 327–335, 1995.

[26] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, pp. 97–109, 1970.

[27] N. Metropolis, A. W. Rosenbluth, M. N Rosenbluth, A. H. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1091, 1953.

[28] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[29] H. Tjelmeland, "Using all metropolis-hastings proposals to estimate mean values.," Tech. Rep., Department of Mathematical Sciences, Norwegian University of Science and Technology,, 2004.

[30] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing*, vol. 22, no. 6, pp. 789–828, Sept. 1996.

[31] W. D. Gropp and E. Lusk, *User's Guide for* `mpich`, *a Portable Implementation of MPI*, Mathematics and Computer Science Division, Argonne National Laboratory, 1996, ANL-96/6.

[32] S. Zhou, R. Chellappa, and B. Moghaddam, "Visual tracking and recognition using appearance-adaptive models in particle filters," *Transactions on Image Processing*, vol. 11, pp. 1434–1456, November 2004.

[33] A.C. Sankaranarayanan, A. Srivastava, and R. Chellappa, "Algorithmic and architectural optimizations for computationally efficient particle filtering," *IEEE Transactions on Image Processing*, vol. 17, no. 5, pp. 737–748, 2008.

[34] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[35] DL Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[36] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[37] E.J. Candčs, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus-Mathématique*, 2008.

[38] D.L. Donoho, "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution," *Communications on Pure and Applied mathematics*, vol. 59, no. 6, pp. 797, 2006.

[39] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied mathematics*, vol. 59, no. 8, pp. 1207, 2006.

[40] S. Ji, Y. Xue, and L. Carin, "Bayesian Compressive Sensing," *IEEE Transactions on Signal Processing*.

[41] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, 2008.

[42] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.

[43] MF Duarte, MA Davenport, D. Takbar, JN Laska, T. Sun, KF Kelly, and RG Baraniuk, "Single-Pixel Imaging via Compressive Sampling [Building simpler, smaller, and less-expensive digital cameras]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[44] V. Cevher, A. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *Proceedings of European Conference on Computer Vision*, 2008, pp. 12–18.

[45] M. Coates, "Distributed particle filters for sensor networks," *Proceedings of the third international symposium on Information processing in sensor networks*, pp. 99–107, 2004.

[46] L. Xiao, S. Boyd, and S.J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[47] X. Sheng, Y. H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," *Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.

[48] S. Saha, S. S. Bhattacharyya, and W. Wolf, "A Communication Interface for Multiprocessor Signal Processing Systems," *IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia*, pp. 127–132, 2006.

[49] L. I. Smith, "A tutorial on Principal Components Analysis," Cornell University, USA, 2002.

[50] A. Hyvrinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, John Wiley and Sons, 2001.

[51] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[52] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," in *Proceedings of the National Academy of Sciences*, 2003, vol. 100, pp. 5591–5596.

[53] J. B. Tenenbaum, V. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[54] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[55] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *SIAM Journal on Scientific Computing*, vol. 26, no. 1, pp. 313–338, 2005.