1

Compressive Sensing in Visual Tracking

Garrett Warnell and Rama Chellappa
University of Maryland, College Park
USA

1. Introduction

Visual tracking is an important component of many video surveillance systems. Specifically, visual tracking refers to the inference of physical object properties (e.g., spatial position or velocity) from video data. This is a well-established problem that has received a great deal of attention from the research community (see, e.g., the survey (Yilmaz et al., 2006)). Classical techniques often involve performing object segmentation, feature extraction, and sequential estimation for the quantities of interest.

Recently, a new challenge has emerged in this field. Tracking has become increasingly difficult due to the growing availability of cheap, high-quality visual sensors. The issue is data deluge (Baraniuk, 2011), i.e., the quantity of data prohibits its usefulness due to the inability of the system to efficiently process it. For example, a video surveillance system consisting of many high-definition cameras may be able to gather data at a high rate (perhaps gigabytes per second), but may not be able to process, store, or transmit the acquired video data under real-time and bandwidth constraints.

The emerging theory of *compressive sensing (CS)* has the potential to address this problem. Under certain conditions related to sparse representations, it effectively reduces the amount of data collected by the system while retaining the ability to faithfully reconstruct the information of interest. Using novel sensors based on this theory, there is hope to accomplish tracking tasks while collecting significantly less data than traditional systems.

This chapter will first present classical components of and approaches to visual tracking, including background subtraction, the Kalman and particle filters, and the mean shift tracker. This will be followed by an overview of CS, especially as it relates to imaging. The rest of the chapter will focus on several recent works that demonstrate the use and benefit of CS in visual tracking.

2. Classical visual tracking

The purpose of this section is to give an overview of classical visual tracking. As a popular component present in many methods, an overview of techniques used for background subtraction will be provided. Next, the focus will shift to the probabilistic tracking frameworks that define the Kalman and particle filters. This will be followed by a presentation of an effective application-specific method: the mean shift tracker.

2.1 Background subtraction

An important first step in many visual tracking systems is the extraction of regions of interest (e.g, those containing objects) from the rest of the scene. These regions are collectively termed the *foreground*, and the technique of *background subtraction* aims to segment it from the background (i.e., the rest of the frame). Once the foreground has been identified, the task of feature extraction becomes much easier due to the resulting decrease in data.

2.1.1 Hypothesis testing formulation

When dealing with digital images, one can pose the problem of background subtraction as a hypothesis test (Poor, 1994; Sankaranarayanan et al., 2008) for each pixel in the image. The null hypothesis (H_0) is that a pixel belongs to the background, while the alternate hypothesis (H_1) is that it belongs to the foreground. Let p denote the measurement observed at an arbitrary pixel. The form of p varies with the sensing modality, however its most common forms are that of a scalar (e.g., light intensity in a gray scale image) or a three-vector (e.g., a color triple in a color image). Whatever they physically represent, let F_B denote the probability distribution over the possible values of p when the pixel belongs to the background, and F_T the distribution for pixels in the foreground. The hypothesis test formulation of background subtraction can then be written as:

$$\begin{aligned} H_0 &: \quad p \sim F_B \\ H_1 &: \quad p \sim F_T \end{aligned} \tag{2.1}$$

The optimal Bayes decision rule for (2.1) is given by:

$$\frac{f_B(p)}{f_T(p)} \underset{H_1}{\overset{H_0}{\gtrless}} \tau \tag{2.2}$$

where $f_B(p)$ and $f_T(p)$ denote the densities corresponding to F_B and F_T respectively, and τ is a threshold determined by the Bayes risk. It is often the case, however, that very little is known about the foreground, and thus the form of F_T. One way of handling this is to assume F_T to be the uniform distribution over the possible values of p. In this case, the above reduces to:

$$f_B(p) \underset{H_1}{\overset{H_0}{\gtrless}} \theta \tag{2.3}$$

where θ is dependent on τ the range of p.

In practice, the optimum value of θ is typically unknown. Therefore, θ is often chosen in an *ad-hoc* fashion such that the decision rule gives pleasing results for the data of interest.

2.1.2 A simple background model

It will now be useful to introduce some notation to handle the temporal and spatial dimensions intrinsic to video data. Let p_i^t denote the value of the i^{th} pixel in the t^{th} frame. Further, let B_i^t parametrize the corresponding background distribution, denoted $F_{B,i,t}$, which may vary with respect to both time and space. In order to select a good hypothesis test, the focus of the background subtraction problem is on how to determine B_i^t from the available data.

An intuitive, albeit naive, approach to this problem is to presume a static background model with respect to time. A common form of this assumption is that $F_{B,i,t}$ is Gaussian with the same covariance for all i. Such a distribution is parametrized only by its mean, and let μ_i specify this value. Substituting the Gaussian density function for $f_B(p)$ in (2.3) yields the following decision rule:

$$\|p_i^t - \mu_i\|_2 \underset{H_1}{\overset{H_0}{\lessgtr}} \eta \tag{2.4}$$

for some threshold η dependent on θ and the covariance. In essence, the above rule amounts to a simple thresholding of the background likelihood function evaluated at the pixel value of interest. This is an intuitive way to perform background subtraction in that if the difference between the background μ_i and the observation p_i^t is high enough, the pixel is classified as belonging to the foreground. Further, this method is computationally advantageous in that it simply requires storing a background image, μ_i for all i, and thresholding the difference between it and a test image. An example of this method is shown in Figure 1.

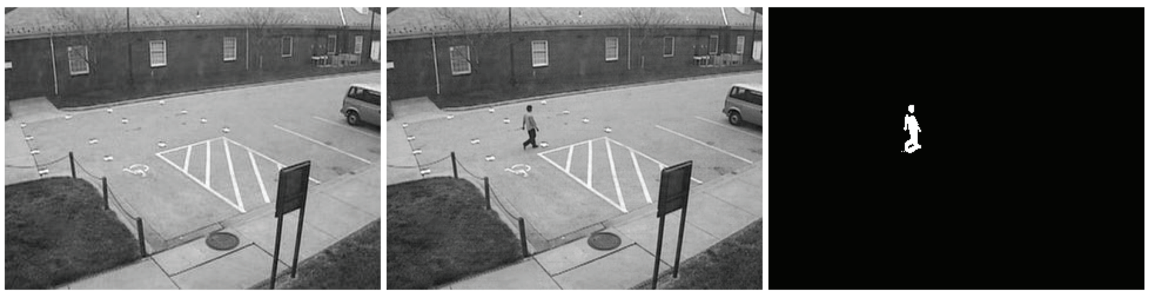

Fig. 1. Background subtraction results for the static unimodal Gaussian model. Left: static background image. Middle: image with human. Right: background subtraction results using the method in (2.4)

2.1.3 Dynamic background modeling

The static approach outlined above is simple, but suffers from the inability to cope with a dynamic background. Such a background is common in video due to illumination shifts, camera and object motion, and other changes in the environment. For example, a tree in the background may sway in the breeze, causing pixel measurements to change significantly from one frame to the next (e.g. tree to sky). However, each shift should not cause the pixel to be classified as foreground, which will occur under the unimodal Gaussian model. A solution to this problem is to use *kernel density estimation (KDE)* (Elgammal et al., 2002; Stauffer & Grimson, 1999) to estimate $f_{B,i,t}$ from past data, i.e.

$$f_{B,i,t}(p) = \frac{1}{N} \sum_{j=t-N}^{t-1} K_j(p) \tag{2.5}$$

where K_j is a kernel density function dependent on the observation p_i^j. For example, K_j may be defined as a Gaussian with fixed covariance and mean p_i^j. Using this definition, B_i^t can be thought of as the pixel history $\{p_i^j\}_{j=t-N}^{t-1}$, and $F_{B,i,t}$ becomes a mixture of Gaussians. This method is also adaptive to temporally recent changes in the background, as only the previous N observations are used in the density estimate.

2.2 Tracking

In general, *tracking* is the sequential estimation of a random variable based on observations over which it exerts influence. In the field of video surveillance, this random variable represents certain physical qualities belonging to objects of interest. For example, Broida and Chellappa (Broida & Chellappa, 1986) characterize a two-dimensional object in the image plane via its center of mass and translational velocity. They also incorporate other quantities to capture shape, global scale, and rotational motion. The time sequential estimates of such quantities are referred to as *tracks*.

To facilitate subsequent discussion, it is useful to consider the discrete time state space representation of the overall system that encompasses object motion and observation. The *state* of the system represents the unknown values of interest (e.g., object position), and in this section it will be denoted by a *state vector*, $\mathbf{x}_t$, whose components correspond to these quantities. Observations of the system will be denoted by $\mathbf{y}_t$, and are obtained via a mapping from the image to the observation space. This process is referred to as *feature extraction*, which will not be the focus of this chapter. Instead, it is assumed that observations are provided to the tracker with some specified probabilistic relationship between observation and state. Given the complicated nature of feature extraction, it is often the case that this relationship is heuristically selected based on some intuition regarding the feature extraction process.

In the context of the above discussion, the goal of a tracker is to provide sequential estimates of $\mathbf{x}_t$ using the observations $(\mathbf{y}_0, \ldots, \mathbf{y}_t)$. In the following sections, a few prominent methods by which this is done will be considered.

2.2.1 Kalman filtering

The Kalman filter is a recursive tracking technique that is widely popular due to its computational efficiency and ease of implementation. Under specific system assumptions, it is able to provide a state estimate that is optimal according to a few popular metrics. This section will outline these assumptions and detail the Kalman filtering method that is used to compute the sequential state estimates.

Specifically, the assumptions that yield optimality are that the physical process governing the behavior of the state should be linear and affected by additive white Gaussian *process noise*, $\mathbf{w}_t$, i.e. (Anderson & Moore, 1979),

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{F}_t \mathbf{x}_t + \mathbf{w}_t \\ \mathbf{w}_t &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t),\ \mathbb{E}\left[\mathbf{w}_k \mathbf{w}_l^T\right] = \mathbf{Q}_k \delta_{kl} \quad , \end{aligned} \tag{2.6}$$

where δ_{kl} is equal to one when $k = l$, and is zero otherwise. The process noise allows for the model to remain valid even when the relationship between $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$ is not completely captured by $\mathbf{F}_t$.

The required relationship between $\mathbf{y}_t$ and $\mathbf{x}_t$ is specified by:

$$\begin{aligned}\mathbf{y}_t &= \mathbf{H}_t^T\mathbf{x}_t + \mathbf{v}_t \\ \mathbf{v}_t &\sim \mathcal{N}(\mathbf{0},\mathbf{R}_t),\ \mathbb{E}\left[\mathbf{v}_k\mathbf{v}_l^T\right] = \mathbf{R}_k\delta_{kl} \quad .\end{aligned} \tag{2.7}$$

Notice that, just as in the state model, the relationship between the observation and the state is assumed to be linear and affected by white Gaussian noise $\mathbf{v}_t$. This is referred to as *measurement noise*, and is assumed to be independent of $\{\mathbf{w}_t\}_{t=0}^{\infty}$.

With the above assumptions, the goal of the Kalman filter is to compute the best estimate of $\mathbf{x}_k$ from the observations $(\mathbf{y}_0,\ldots,\mathbf{y}_t)$. What is meant by "best" can vary from application to application, but common criterion yield the *maximum a posteriori (MAP)* and *minimum mean squared error (MMSE)* estimators. Regardless of the estimator chosen, the value it yields can be computed using the posterior density $p(\mathbf{x}_t|\mathbf{y}_0,\ldots,\mathbf{y}_t)$. For example, the MMSE estimate is the mean of this density and the MAP estimate is the value of $\mathbf{x}_t$ that maximizes it.

Under the assumptions made when specifying the state and observation equations, the MMSE and MAP estimates are identical. Since successive estimates can be calculated recursively, the Kalman filter provides this estimate without having to re-compute $p(\mathbf{x}_t|\mathbf{y}_0,\ldots,\mathbf{y}_t)$ each time a new observation is received. This benefit requires the additional assumption that $\mathbf{x}_0 \sim \mathcal{N}(\bar{\mathbf{x}}_0,\mathbf{P}_0)$, which is equivalent to assuming $\mathbf{x}_0$ and $\mathbf{y}_0$ to be jointly Gaussian, i.e.,

$$\begin{bmatrix}\mathbf{x}_0\\ \mathbf{y}_0\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\bar{\mathbf{x}}_0\\ \mathbf{H}_0^T\bar{\mathbf{x}}_0\end{bmatrix}, \begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_0\mathbf{H_0}\\ \mathbf{H}_0^T\mathbf{P}_0 & \mathbf{H}_0^T\mathbf{P}_0\mathbf{H}_0+\mathbf{R}_0\end{bmatrix}\right) \quad , \tag{2.8}$$

which yields

$$\begin{aligned}\mathbf{x}_0|\mathbf{y}_0 &\sim \mathcal{N}\left(\hat{\mathbf{x}}_{0|0},\boldsymbol{\Sigma}_{0|0}\right) & (2.9)\\ \hat{\mathbf{x}}_{0|0} &= \bar{\mathbf{x}}_0 + \mathbf{P}_0\mathbf{H}_0(\mathbf{H}_0^T\mathbf{P}_0\mathbf{H}_0+\mathbf{R}_0)^{-1}(\mathbf{y}_0-\mathbf{H}_0^T\bar{\mathbf{x}}_0) & \\ \boldsymbol{\Sigma}_{0|0} &= \mathbf{P}_0 - \mathbf{P}_0\mathbf{H}_0(\mathbf{H}_0^T\mathbf{P}_0\mathbf{H}_0+\mathbf{R}_0)^{-1}\mathbf{H}_0^T\mathbf{P}_0 \quad . & (2.10)\end{aligned}$$

Since $\mathbf{x}_0|\mathbf{y}_0$ is Gaussian, both its MMSE and MAP estimates are given by the mean of this distribution, i.e., $\hat{\mathbf{x}}_{0|0}$. The subscript indicates that this is the estimate of $\mathbf{x}_0$ given observations up to time 0.

From this starting point, the Kalman filter calculates subsequent estimates ($\hat{\mathbf{x}}_{t|t}$ in general) using a two step procedure. First, it can be seen that $\mathbf{x}_{t+1}|\mathbf{y}_{0:t}$ is also Gaussian, with mean and covariance given by

$$\begin{aligned}\hat{\mathbf{x}}_{t+1|t} &= \mathbf{F}_t\hat{\mathbf{x}}_{t|t} \\ \boldsymbol{\Sigma}_{t+1|t} &= \mathbf{F}_t\boldsymbol{\Sigma}_{t|t}\mathbf{F}_t^T + \mathbf{Q}_t \quad .\end{aligned} \tag{2.11}$$

The above are known as the *time update equations*. Once $\mathbf{y}_{t+1}$ is observed, the second step of the Kalman filter is to adjust the prediction $\hat{\mathbf{x}}_{t+1|t}$ to one that incorporates the information provided by the new observation. This is done via the *measurement update equations*:

$$\hat{\mathbf{x}}_{t+1|t+1} = \hat{\mathbf{x}}_{t+1|t} + \mathbf{\Sigma}_{t+1|t}\mathbf{H}_{t+1}(\mathbf{H}_{t+1}^{T}\mathbf{\Sigma}_{t+1|t}\mathbf{H}_{t+1} + \mathbf{R}_{t+1})^{-1}(\mathbf{y}_{t+1} - \mathbf{H}_{t+1}^{T}\hat{\mathbf{x}}_{t+1|t}) \tag{2.12}$$

$$\mathbf{\Sigma}_{t+1|t+1} = \mathbf{\Sigma}_{t+1|t} - \mathbf{\Sigma}_{t+1|t}\mathbf{H}_{t+1}(\mathbf{H}_{t+1}^{T}\mathbf{\Sigma}_{t+1|t}\mathbf{H}_{t+1} + \mathbf{R}_{t+1})^{-1}\mathbf{H}_{t+1}^{T}\mathbf{\Sigma}_{t+1|t} \quad . \tag{2.13}$$

Using the above steps at each time instant, the Kalman filter provides optimal tracks $\{\hat{\mathbf{x}}_{t|t}\}_{t=0}^{\infty}$ that are calculated in a recursive and efficient manner. The optimality of the estimates comes at the cost of requiring the assumptions of linearity and Gaussianity in the state space formulation of the system. Even without the Gaussian assumptions, the filter is optimal among the class of linear filters.

2.2.2 Particle filtering

Since it is able to operate in an unconstrained setting, the *particle filter* (Doucet et al., 2001; Isard & Blake, 1996) is a more general approach to sequential estimation. However, this expanded utility comes at the cost of high computational complexity. The particle filter is a *sequential Monte Carlo method*, using samples of the conditional distribution in order to approximate it and thus the desired estimates. There are many variations of the particle filter, but the focus of this section shall be on the so-called *bootstrap filter*.

Assume the system of interest behaves according to the following known densities:

$$p(\mathbf{x}_0) \quad , \tag{2.14}$$

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad t \geq 1 \quad \text{, and} \tag{2.15}$$

$$p(\mathbf{y}_t|\mathbf{x}_t), \quad t \geq 1 \quad . \tag{2.16}$$

Note that the more general specifications $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$ replace the linear, Gaussian descriptions of the system and observation behaviors necessary for the Kalman filter. In order to achieve the goal of tracking, it is necessary to have some information regarding $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ (from which $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is apparent), where $\mathbf{x}_{0:t} = (\mathbf{x}_0, \ldots, \mathbf{x}_t)$, and similarly for $\mathbf{y}_{1:t}$. Here, we depart from the previous notation and assume that the first observation is available at $t = 1$.

In a purely Bayesian sense, one could compute the conditional density as

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})d\mathbf{x}_{0:t}} \quad , \tag{2.17}$$

which leads to a recursive formula

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})\frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{y}_t|\mathbf{y}_{t-1})} \quad . \tag{2.18}$$

A similar type of recursion can be shown to exist for the marginal density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. While the above expressions seem simple, for general distributions in (2.14) (2.15) and (2.16), they often become prohibitively difficult to evaluate due to analytic and computational complexity.

The particle filter avoids the analytic difficulties above using Monte Carlo sampling. If N i.i.d. *particles* (samples), $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^{N}$, drawn from $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ were available, one could approximate the density by placing a Dirac delta mass at the location of each sample, i.e.,

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx P_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{(i)}) \quad . \tag{2.19}$$

It would then be straightforward to use P_N to calculate an estimate of the random variable (i.e. a track). However, this method presents its own difficulty in that it is usually impractical to obtain the samples $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^{N}$.

The bootstrap filter is based on a technique called *sequential importance sampling*, which is used to overcome the issue above. Samples are initially drawn from the known prior distribution $p(\mathbf{x}_0)$, from which it is straightforward to generate samples $\{\mathbf{x}_0^{(i)}\}_{i=1}^{N}$. Next, importance sampling occurs. First, a prediction step takes place, generating candidate samples $\{\tilde{\mathbf{x}}_1^{(i)}\}_{i=1}^{N}$ by drawing $\tilde{\mathbf{x}}_1^{(i)}$ from $p(\mathbf{x}_1|\mathbf{x}_0^{(i)})$ for each i. From here, *importance weights* $\tilde{w}_1^{(i)} = p(\mathbf{y}_1|\tilde{\mathbf{x}}_1^{(i)})$ are calculated based on the observation $\mathbf{y}_1$ and adjusted such that they are normalized (i.e. such that $\sum_i \tilde{w}_1^{(i)} = 1$). The filter then enters the selection step, where samples $\{\mathbf{x}_1^{(i)}\}_{i=1}^{N}$ are generated via draws from a discrete distribution over $\{\tilde{\mathbf{x}}_1^{(i)}\}_{i=1}^{N}$ with the probability for the i^{th} element given by $\tilde{w}_1^{(i)}$. This process is then repeated to obtain $\{\mathbf{x}_2^{(i)}\}_{i=1}^{N}$ from $\{\mathbf{x}_1^{(i)}\}_{i=1}^{N}$ and $\mathbf{y}_2$, and so forth.

Due to the selection step, those candidate particles $\tilde{\mathbf{x}}_t^{(i)}$ for which $p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{i})$ is low will not propagate to the next stage. The samples that survive are those that explain the data well, and are thus concentrated in the most dense areas of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. Therefore, the computed value for common estimators such as the mean and mode will be good approximations of their actual values. Further, note that the candidate particles are drawn from $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, which introduces process noise to prevent the particles from becoming too short-sighted.

Using the estimate calculated from the density approximation yielded by the particles $\{\mathbf{x}_t^{(i)}\}_{i=1}^{N}$, the particle filter is able provide tracks that are optimal for a wide variety of criteria in a more general setting than that required by the Kalman filter. However, the validity of the track depends on the ability of the particles to sufficiently characterize the underlying density. Often, this may require a large number of particles, which can lead to a high computational cost.

2.2.3 Mean shift tracking

Unlike the Kalman and particle filters, the *mean shift* tracker (Comaniciu et al., 2003) is a procedure designed specifically for visual data. The feature employed, a spatially weighted color histogram, is computed directly from the input images. The estimate for the object position in the image plane is defined as the mode of a density over spatial locations, where this density is defined using a similarity measure between the histogram for an object model (i.e. a “template") and the histogram at a location of interest. The mean shift procedure (Comaniciu & Meer, 2002) is then used to find this mode.

In general, the mean shift procedure provides a way to perform gradient ascent on an unknown density using only samples generated by this density. It achieves this via selecting a specific method of density estimation and analytically deriving a data-dependent term that corresponds to the gradient of the estimate. This term is known as the mean shift, and it can be used as the step term in a mode-seeking gradient ascent procedure. Specifically, non-parametric KDE is employed, i.e.,

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \quad , \tag{2.20}$$

where the d-dimensional vector $\mathbf{x}$ represents the feature, $\hat{f}(\cdot)$ the estimated density, and $K(\cdot)$ a *kernel function*. The kernel function is assumed to be radially symmetric, i.e., $K(\mathbf{x}) = c_{k,d}k(\|\mathbf{x}\|^2)$ for some function $k(\cdot)$ and normalizing constant $c_{k,d}$. Using this in (2.20), $\hat{f}(\mathbf{x})$ becomes

$$\hat{f}_{h,K}(\mathbf{x}) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k(\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\|^2) \quad . \tag{2.21}$$

Ultimately, it is the gradient of this approximation, $\nabla \hat{f}_{h,K}$, that is of interest. Letting $g(\cdot) = -k'(\cdot)$, it is given by

$$\nabla \hat{f}_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x}\right] \quad . \tag{2.22}$$

Using $g(\cdot)$ to define a new kernel $G(\mathbf{x}) = c_{g,d}g(\|\mathbf{x}\|^2)$, (2.22) can be rewritten as

$$\nabla \hat{f}_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{n^2 c_{g,d}} \hat{f}_{h,G}(\mathbf{x})\mathbf{m}_{h,G}(\mathbf{x}) \quad , \tag{2.23}$$

where $\mathbf{m}_{h,G}(\mathbf{x})$ denotes the mean shift:

$$\mathbf{m}_{h,G}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x}\right] \quad . \tag{2.24}$$

It can be seen from (2.23) that $\mathbf{m}_{h,G}(\mathbf{x})$ is proportional to $\nabla \hat{f}_{h,K}(\mathbf{x})$, and thus may be used as a step direction in a gradient ascent procedure to find a maximum of $\hat{f}_{h,K}(\mathbf{x})$ (i.e., a mode).

(Comaniciu et al., 2003) utilize the above procedure when tracking objects in the image plane. The selected feature is a spatially weighted color histogram computed over a normalized window of finite spatial support. The spatial weighting is defined by an isotropic kernel $k(\cdot)$, and the object model is given by an m-bin histogram $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1}^{m}$, where

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)\delta\left[b(\mathbf{x}_i^*) - u\right] \quad . \tag{2.25}$$

$\mathbf{x}_i^*$ denotes the spatial location of the i^{th} pixel in the n pixel window containing the object model, assuming the center of the window to be located at $\mathbf{0}$. $\delta\left[b(\mathbf{x}_i^* - u\right]$ is 1 when the pixel value at $\mathbf{x}_i^*$ falls into the u^{th} bin of the histogram, and 0 otherwise. Finally, C is a normalizing constant to ensure that $\mathbf{q}$ is a true histogram.

An object candidate feature located at position $\mathbf{y}$ is denoted by $\hat{\mathbf{p}}(\mathbf{y})$, and is calculated in a manner similar to $\hat{\mathbf{q}}$, except $k(\|\mathbf{x}_i^*\|^2)$ is replaced by $k(\|\mathbf{y} - \mathbf{x}_i\|^2)$ to account for the new window location.

To capture a notion of similarity between $\hat{\mathbf{p}}(\mathbf{y})$ and $\hat{\mathbf{q}}$, the Bhattacharyya coefficient is used, i.e.,

$$d(\mathbf{y}) = \sqrt{1 - \hat{\rho}(\mathbf{y})} \quad , \tag{2.26}$$

where $\hat{\rho}(\mathbf{y}) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u}$ is the Bhattacharyya coefficient.

An approximation of $\hat{\rho}(\mathbf{y})$ is provided by

$$\hat{\rho}(\mathbf{y}) = \frac{1}{2}\sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y}_0)\hat{q}_u} + \frac{C_h}{2}\sum_{i=1}^{n} w_i k\left(\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\|^2\right) \quad . \tag{2.27}$$

Above, $\mathbf{y}_0$ represents an initial location provided by the track from the previous frame. The weights $\{w_i\}_{i=1}^n$ are calculated as a function of $\hat{\mathbf{q}}$, $\hat{\mathbf{p}}(\mathbf{y}_0)$, and $b(\mathbf{x}_i)$. To minimize the distance in (2.26), the second term of (2.27) should be maximized with respect to $\mathbf{y}$. This term can be interpreted as a nonparametric weighted KDE with kernel function $k(\cdot)$. Thus, the mean shift procedure can be used to iterate over $\mathbf{y}$ and find that value which minimizes $d(\mathbf{y})$. The result is then taken to be the location estimate (track) for the current frame.

2.3 The data challenge

Given the above background, it can be seen how large amounts of data can be of detriment to tracking. Background subtraction techniques may require complicated density estimates for each pixel, which become burdensome in the presence of high-resolution imagery. The filtering methods presented above are not specific to the amount of data, but more of it leads to greater computational complexity when performing the estimation. Likewise, higher data dimensionality is of detriment to mean shift tracking, specifically during the required density estimation and mode search. This extra data could be due to higher sensor resolution or perhaps the presence of multiple sensors (Sankaranarayanan et al., 2008)(Sankaranarayanan & Chellappa, 2008). Therefore, new tracking strategies must be developed. The hope for finding such strategies comes from the fact that there is a substantial difference in the amount of data collected by these systems compared to the quantity of information that is ultimately of use. Compressive sensing provides a new perspective that radically changes the sensing process with the above observation in mind.

3. Compressive sensing

Compressive sensing is an emerging theory that allows for a certain class of discrete signals to be adequately sensed using far fewer measurements than the dimension of the ambient space in which they reside. By "adequately sensed," it is meant that the signal of interest can be accurately inferred from the measurements collected during the sensing process. In the context of imaging, consider an unknown $n \times n$ grayscale image $\mathbf{F}$, i.e., $\mathbf{F} \in \mathbb{R}^{n \times n}$. A traditional camera measures $\mathbf{F}$ using an $n \times n$ array of photodetectors, where the measurement collected at each detector corresponds to a single pixel value in $\mathbf{F}$. If $\mathbf{F}$ is vectorized as $\mathbf{x} \in \mathbb{R}^N$ $(N = n^2)$, then the imaging strategy described above amounts to (in the noiseless case) $\hat{\mathbf{x}} = \mathbf{y} = \mathbf{I}\mathbf{x}$ (Romberg, 2008), where $\hat{\mathbf{x}}$ is the inferred value of $\mathbf{x}$ using the measurements $\mathbf{y}$. Each component of $\mathbf{y}$ (i.e., a measurement) corresponds to a single component of $\mathbf{x}$, and this relationship is captured by representing the sensing process as the identity matrix $\mathbf{I}$. Since $\mathbf{x}$ is the quantity of interest, estimating it from $\mathbf{y}$ also amounts to a simple identity mapping, i.e. $\hat{\mathbf{x}}(\mathbf{y}) = \mathbf{y}$. However, both the measurement and estimation process can change, giving rise to interesting and useful signal acquisition methodologies.

For practical purposes, it is often the case that $\mathbf{x}$ is represented using far fewer measurements than the N collected above. For example, using *transform coding* methods (e.g., JPEG 2000), $\mathbf{x}$ can usually be closely approximated by specifying very few values compared to N (Bruckstein et al., 2009). This is accomplished via obtaining $\mathbf{b} = \mathbf{B}\mathbf{x}$ for some orthonormal basis $\mathbf{B}$ (e.g., the wavelet basis), and setting all but the k largest components of $\mathbf{b}$ to zero. If this new vector is denoted $\mathbf{b}_k$, then the transform coding approximation of $\mathbf{x}$ is given by $\hat{\mathbf{x}} = \mathbf{B}^{-1}\mathbf{b}_k$. If $\|\mathbf{x} - \hat{\mathbf{x}}\|_2$ is small, then this approximation is a good one. Since $\mathbf{B}$ is orthonormal, this condition also requires that $\|\mathbf{b} - \mathbf{b}_k\|_2$ be small as well. If such is the case, $\mathbf{b}$ is said to be *k-sparse* (and $\mathbf{x}$ *k-sparse in* $\mathbf{B}$), i.e., most of the energy in $\mathbf{b}$ is distributed among very few of its components. Thus, if the value of $\mathbf{x}$ is known, and $\mathbf{x}$ is k-sparse in $\mathbf{B}$, a good approximation of $\mathbf{x}$ can be obtained from $\mathbf{b}_k$. Compression comes about since $\mathbf{b}_k$ (and thus $\mathbf{x}$) can be specified using just $2k$ quantities instead of N: the values and locations of the k largest coefficients in $\mathbf{b}$. However, extracting such information requires full knowledge of $\mathbf{x}$, which necessitates N measurements using the traditional imaging system above. Thus, N data points must be collected when in essence all but $2k$ are thrown away. This is not completely unjustified, as one cannot hope to form $\mathbf{b}_k$ without knowing $\mathbf{b}$. On the other hand, such a large disparity between the amount of data collected and the amount that is truly useful seems wasteful.

This glaring disparity is what CS seeks to address. Instead of collecting N measurements of $\mathbf{x}$, the CS strategy is to collect M, where $M << N$ and depends on k. As long as $\mathbf{x}$ is k-sparse in some basis and an appropriate decoding procedure is employed, these M values yield a good approximation of $\mathbf{x}$. For example, let $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ be the *measurement matrix* by which these values, $\mathbf{y} \in \mathbb{R}^M$, are obtained as $\mathbf{y} = \mathbf{\Phi}\mathbf{x}$. Further, assume $\mathbf{x}$ is k-sparse. It is possible to recover $\mathbf{x}$ from $\mathbf{y}$ if $\mathbf{\Phi}$ has the *restricted isometry property (RIP)* of order $2k$ (Candès & Wakin, 2008), i.e., the smallest δ for which

$$(1-\delta) \leq \frac{\|\mathbf{\Phi}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} \leq (1+\delta) \tag{3.1}$$

holds for all $2k$-sparse vectors is not too close to 1. An intuitive interpretation of this property is that it ensures that all $2k$-sparse vectors do not lie in Null($\mathbf{\Phi}$). This guarantees that a unique measurement $\mathbf{y}$ is generated for each k-sparse $\mathbf{x}$ even though $\mathbf{\Phi}$ is underdetermined.

An example $\mathbf{\Phi}$ that satisfies the above conditions is one for which entries are drawn from the Bernoulli distribution over the discrete set $\{\frac{-1}{\sqrt{N}}, \frac{1}{\sqrt{N}}\}$ and each realization is equally likely (Baraniuk, 2007). If, in addition, M is selected such that $M > Ck \log N$ for a specific constant

C, it is overwhelmingly likely that $\mathbf{\Phi}$ will be $2k$-RIP. There are other constructions that provide similar guarantees given slightly different bounds on M, but the concept remains unchanged: if M is "large enough," $\mathbf{\Phi}$ will exhibit the RIP with overwhelming probability. Given such a matrix, and considering that this implies a unique $\mathbf{y}$ for each k-sparse $\mathbf{x}$, an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ is ideally calculated from $\mathbf{y}$ as

$$\hat{\mathbf{x}} = \min_{\mathbf{z}\in\mathbb{R}^N} \|\mathbf{z}\|_0 \quad \text{subject to} \quad \mathbf{\Phi z} = \mathbf{y} \quad , \tag{3.2}$$

where $\|\cdot\|_0$, referred to as the ℓ_0 "norm," counts the number of nonzero entries in $\mathbf{z}$. Thus, (3.2) seeks the sparsest vector that explains the observation $\mathbf{y}$. In practice, (3.2) is not very useful since the program it specifies has combinatorial complexity. However, this problem is also mitigated due to the special construction of $\mathbf{\Phi}$ and the fact that $\mathbf{x}$ is k-sparse. Under these conditions, the solution of the following program yields the same results as (3.2) with overwhelming probability:

$$\hat{\mathbf{x}} = \min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{z}\|_1 \quad \text{subject to} \quad \mathbf{\Phi z} = \mathbf{y} \quad . \tag{3.3}$$

Thus, by modifying the sensor to use $\mathbf{\Phi}$ and the decoder to use (3.3), $M << N$ measurements of a k-sparse $\mathbf{x}$ suffice to retain the ability to reconstruct it.

Sensors based on the above theory are beginning emerge (Willett et al., 2011). One of the most notable is the single pixel camera (Duarte et al., 2008), where measurements specified by each row of $\mathbf{\Phi}$ are sequentially computed in the optical domain via a digital micromirror device and a single photodiode. Many of the strategies discussed in the following section assume that the tracking system is such that these compressive sensors replace more traditional cameras.

4. Compressive sensing in video surveillance

Compressive sensing can help alleviate some of the challenges associated with performing classical tracking in the presence of overwhelming amounts of data. By replacing traditional cameras with compressive sensors or by making use of CS techniques in other areas of the process, the amount of data that the system must handle can be drastically reduced. However, this capability should not come at the cost of a significant decrease in tracking performance. This section will present a few methods for performing various tracking tasks that take advantage of CS in order to reduce the quantity of data that must be processed. Specifically, recent methods using CS to perform background subtraction, more general signal tracking, multi-view visual tracking, and particle filtering will be discussed.

4.1 Compressive sensing for background subtraction

One of the most intuitive applications of compressive sensing in visual tracking is the modification of background subtraction such that it is able to operate on compressive measurements. As mentioned in Section 2.1, background subtraction aims to segment the object-containing foreground from the uninteresting background. This process not only helps to localize objects, but also reduces the amount of data that must be processed at later stages of tracking. However, traditional background subtraction techniques require that the full image be available before the process can begin. Such a scenario is reminiscent of the problem that

CS aims to address. Noting that the foreground signal (image) is sparse in the spatial domain, (Cevher et al., 2008) have presented a technique via which background subtraction can be performed on compressive measurements of a scene, resulting in a reduced data rate while simultaneously retaining the ability to reconstruct the foreground. More recently, (Warnell et al., 2012) have proposed a modification to this technique which adaptively adjusts the number of compressive measurements collected to the dynamic foreground sparsity typical to surveillance data.

Denote the images comprising a video sequence as $\{\mathbf{x}_t\}_{t=0}^{\infty}$, where $\mathbf{x}_t \in \mathbb{R}^N$ is the vectorized image captured at time t. Cevher et al. model each image as the sum of foreground and background components $\mathbf{f}_t$ and $\mathbf{b}_t$, respectively. That is,

$$\mathbf{x}_t = \mathbf{f}_t + \mathbf{b}_t \quad . \tag{4.1}$$

Assume $\mathbf{x}_t$ is sensed using $\mathbf{\Phi} \in \mathbb{C}^{M \times N}$ to obtain compressive measurements $\mathbf{y}_t = \mathbf{\Phi}\mathbf{x}_t$. If $\Delta(\mathbf{\Phi}, \mathbf{y})$ represents a CS decoding procedure such as (3.3), then the proposed method for estimating $\mathbf{f}_t$ from $\mathbf{y}_t$ is

$$\hat{\mathbf{f}}_t = \Delta(\mathbf{\Phi}, \mathbf{y} - \mathbf{y}_t^b) \quad , \tag{4.2}$$

where it is assumed that $\mathbf{y}_t^b = \mathbf{\Phi}\mathbf{b_t}$ is known via an estimation and update procedure.

To begin, $\mathbf{y}_0^b$ is initialized using a sequence of N compressively sensed background-only frames $\{\mathbf{y}_j^b\}_{j=1}^N$ that appear before the sequence of interest begins. These measurements are assumed to be realizations of a multivariate Gaussian random variable, and the maximum likelihood (ML) procedure is used to estimate its mean as $\mathbf{y}_0^b = \frac{1}{N}\sum_{j=1}^N \mathbf{y}_j^b$. This estimate is used as the known background for $t = 0$ in (4.2). Since the background typically changes over time, a method is proposed for updating the background estimate based on previous observations. Specifically, the following is proposed:

$$\mathbf{y}_{t+1}^b = \alpha(\mathbf{y}_t - \mathbf{\Phi}\Delta(\mathbf{\Phi}, \mathbf{y}_{t+1}^{ma})) + (1-\alpha)\mathbf{y}_t^b \tag{4.3}$$

$$\mathbf{y}_{t+1}^{ma} = \gamma\mathbf{y}_t + (1-\gamma)\mathbf{y}_t^{ma} \quad , \tag{4.4}$$

where $\alpha, \gamma \in (0,1)$ are learning rate parameters and $\mathbf{y}_{t+1}^{ma}$ is a moving average term. This method compensates for both gradual and sudden changes to the background. A block diagram of the proposed system is shown in Figure 2.

The above procedure assumes a fixed $\mathbf{\Phi} \in \mathbb{C}^{M \times N}$. Therefore, M compressive measurements of $\mathbf{x}_t$ are collected at time t regardless of its content. It is not hard to imagine that the number of significant components of $\mathbf{f}_t$, k_t, might vary widely with t. For example, consider a scenario in which the foreground consists of a single object at $t = t_0$, but many more at $t = t_1$. Then $k_1 > k_0$, and $M > Ck_1 \log N$ implies that $\mathbf{x}_{t_0}$ has been oversampled due to the fact that only $M > Ck_0 \log N$ measurements are necessary to obtain a good approximation of $\mathbf{f}_{t_0}$. Foregoing the ability to update the background, (Warnell et al., 2012) propose a modification to the above method for which the number of compressive measurements at each frame, M_t, can vary.

Such a scheme requires a different measurement matrix for each time instant, i.e. $\mathbf{\Phi_t} \in \mathbb{C}^{M_t \times N}$. To form $\mathbf{\Phi_t}$, one first constructs $\mathbf{\Phi} \in \mathbb{C}^{N \times N}$ via standard CS measurement matrix

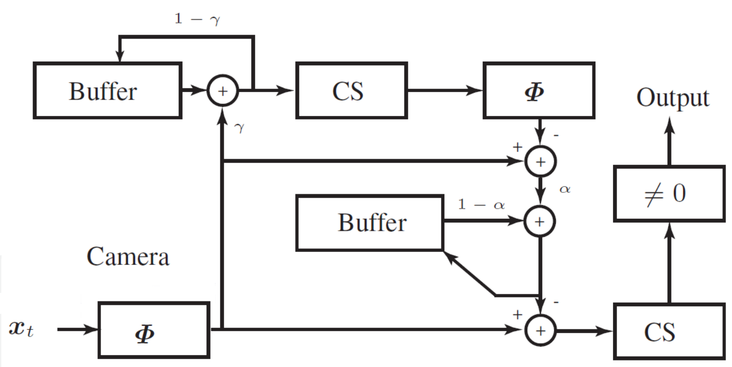

Fig. 2. Block diagram of the compressive sensing for background subtraction technique. Figure originally appears in (Cevher et al., 2008).

construction techniques. $\mathbf{\Phi_t}$ is then formed by selecting only the first M_t rows of $\mathbf{\Phi}$ and column-normalizing the result. The fixed background estimate, $\mathbf{y}^b$, is estimated from a set of measurements of the background only obtained via $\mathbf{\Phi}$. In order to use this estimate at each time instant t, $\mathbf{y}_t^b$ is formed by retaining only the first M_t components of $\mathbf{y}^b$.

In parallel to $\mathbf{\Phi}_t$, the method also requires an extra set of compressive measurements via which the quality of the foreground estimate, $\hat{\mathbf{f}}_t = \Delta(\mathbf{\Phi}_t, \mathbf{y}_t - \mathbf{y}_t^b)$, is determined. These are obtained via a *cross validation* matrix $\mathbf{\Psi} \in \mathbb{C}^{r\times N}$, which is constructed in a manner similar to $\mathbf{\Phi}$. r depends on the desired accuracy of the cross validation error estimate (given below), is negligible compared to N, and constant for all t. In order to use the measurements $\mathbf{z}_t = \mathbf{\Psi}\mathbf{x}_t$, it is necessary to perform background subtraction in this domain via an estimate of the background, $\mathbf{z}^b$, which is obtained in a manner similar to $\mathbf{y}^b$ above.

The quality of $\hat{\mathbf{f}}_\mathbf{t}$ depends on the relationship between k_t and M_t. Using a technique operationally similar to cross validation, an estimate of $\|\mathbf{f}_t - \hat{\mathbf{f}}_t\|_2$, i.e., the error between the true foreground and the reconstruction provided by Δ at time t, is provided by $\|(\mathbf{z}_t - \mathbf{z}^b) - \mathbf{\Psi}\hat{\mathbf{f}}_t\|_2$. M_{t+1} is set to be greater or less than M_t depending on the hypothesis test

$$\|(\mathbf{z}_t - \mathbf{z}^b) - \mathbf{\Psi}\hat{\mathbf{f}}_t\|_2 \lesseqgtr \tau_t \quad . \tag{4.5}$$

Here, τ_t is a quantity set based on the expected value of $\|\mathbf{f}_t - \hat{\mathbf{f}}_t\|_2$ assuming M_t to be large enough compared to k_t. The overall algorithm is termed *adaptive rate compressive sensing (ARCS)*, and the performance of this method compared to a non-adaptive approach is shown in Figure 3.

Both techniques assume that the tracking system can only collect compressive measurements and provide a method by which foreground images can be reconstructed. These foreground images can then be used just as in classical tracking applications. Thus, CS has provided a means by which to reduce the up-front data costs associated with the system while retaining the information necessary to track.

4.2 Kalman filtered compressive sensing

A more general problem regarding signal tracking using compressive observations is considered in (Vaswani, 2008). The signal being tracked, $\{\mathbf{x}_t\}_{t=0}^{\infty}$, is assumed to be both sparse

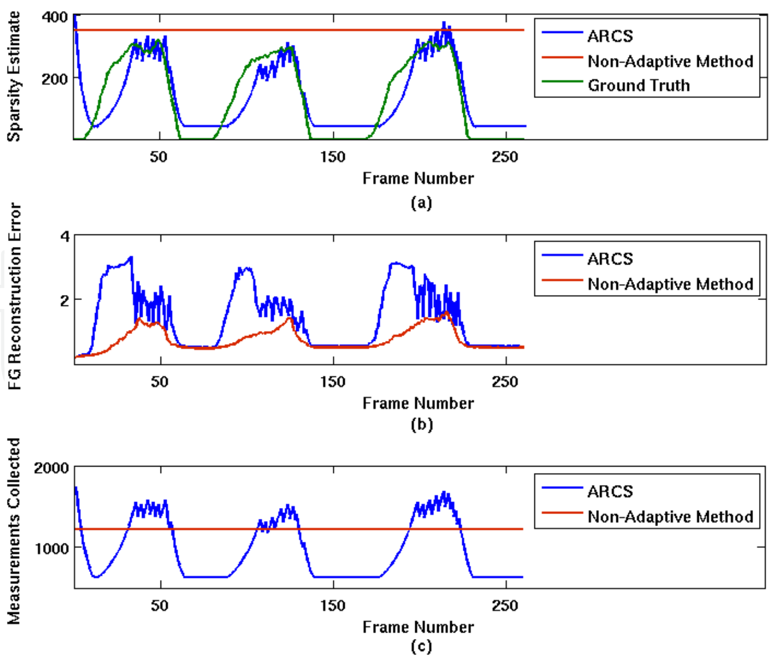

Fig. 3. Comparison between ARCS and a non-adaptive method for a dataset consisting of vehicles moving in and out of the field of view. (a) Foreground sparsity estimates for each frame, including ground truth. (b) ℓ_2 foreground reconstruction error. (c) Number of measurements required. Note the measurements savings provided by ARCS for most frames, and its ability to track the dynamic foreground sparsity. Figure originally appears in (Warnell et al., 2012).

and have a slowly-changing sparsity pattern. Given these assumptions, if the support set of $\mathbf{x}_t$, T_t, is known, the relationship between $\mathbf{x}_t$ and $\mathbf{y}_t$ can be written as:

$$\mathbf{y}_t = \mathbf{\Phi}_{T_t}(\mathbf{x})_{T_t} + \mathbf{w}_t \quad . \tag{4.6}$$

Above, $\mathbf{\Phi}$ is the CS measurement matrix, and $\mathbf{\Phi}_{T_t}$ retains only those columns of $\mathbf{\Phi}$ whose indices lie in T_t. Likewise, $(\mathbf{x}_t)_{T_t}$ contains only those components corresponding to T_t. Finally, $\mathbf{w}_t$ is assumed to be zero mean Gaussian noise. If $\mathbf{x}_t$ is assumed to also follow the state model $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_t$ with $\mathbf{v}_t$ zero mean Gaussian noise, then the MMSE estimate of $\mathbf{x}_t$ from $\mathbf{y}_t$ can be computed using a Kalman filter instead of a CS decoder.

The above is only valid if T_t is known, which is often not the case. This is handled by using the Kalman filter output to detect changes in T_t and re-estimate it if necessary. $\tilde{\mathbf{y}}_{t,f} = \mathbf{y}_t - \mathbf{\Phi}\hat{\mathbf{x}}$, the filter error, is used to detect changes in the signal support via a likelihood ratio test given by

$$\tilde{\mathbf{y}}'_{t,f}\mathbf{\Sigma}\tilde{\mathbf{y}}_{t,f} \gtrless \tau \tag{4.7}$$

where τ is a threshold and $\mathbf{\Sigma}$ is the filtering error covariance. If the term on the left hand side exceeds the threshold, then changes to the support set are found by applying a procedure based on the Dantzig selector. Once T_t has been re-estimated, $\hat{\mathbf{x}}$ is re-evaluated using this new support set.

The above algorithm is useful in surveillance scenarios when objects under observation are stationary or slowly-moving. Under such assumptions, this method is able to perform signal tracking with a low data rate and low computational complexity.

4.3 Joint compressive video coding and analysis

(Cossalter et al., 2010) consider a collection of methods via which systems utilizing compressive imaging devices can perform visual tracking. Of particular note is a method referred to as *joint compressive video coding and analysis*, via which the tracker output is used to improve the overall effectiveness of the system. Instrumental to this method is work from theoretical CS literature which proposes a weighted decoding procedure that iteratively determines the locations and values of the (nonzero) sparse vector coefficients. Modifying this decoder, the joint coding and analysis method utilizes the tracker estimate to directly influence the weights. The result is a foreground estimate of higher quality compared to one obtained via standard CS decoding techniques.

The weighted CS decoding procedure calculates the foreground estimate via

$$\hat{\mathbf{f}} = \min_{\theta} \|\mathbf{W}\theta\|_1 \quad \text{s.t.} \quad \|\mathbf{y}^f - \mathbf{\Phi}\theta\|_2 \leq \sigma \quad , \tag{4.8}$$

where $\mathbf{y}^f = \mathbf{y} - \mathbf{y}^b$, $\mathbf{W}$ is a diagonal matrix with weights $[w(1) \ldots w(N)]$, and σ captures the expected measurement and quantization noise in $\mathbf{y}^f$. Ideally, the weights are selected according to

$$w(i) = \frac{1}{|f(i)| + \epsilon} \quad , \tag{4.9}$$

where $f(i)$ is the value of the i^{th} coefficient in the true foreground image. Of course, these values are not known in advance, but the closer the weights are to their actual value, the more accurate $\hat{\mathbf{f}}$ becomes. The joint coding and analysis approach utilizes the tracker output in selecting appropriate values for these weights.

The actual task of tracking is accomplished using a particle filter similar to that presented in Section 2.2.2. The state vector for an object at time t is denoted by $\mathbf{z}_t = [\mathbf{c}_t\ \mathbf{s}_t\ \mathbf{u}_t]$, where $\mathbf{s}_t$ represents the size of the bounding box defined by the object appearance, $\mathbf{c}_t$ the centroid of this box, and $\mathbf{u}_t$ the object velocity in the image plane. A suitable kinematic motion model is utilized to describe the expected behavior of these quantities with respect to time, and foreground reconstructions are used to generate observations.

Assuming the foreground reconstruction $\hat{\mathbf{f}}_t$ obtained via decoding the compressive observations from time t is accurate, a reliable tracker estimate can be computed. This estimate, $\hat{\mathbf{z}}_t$, can then be used to select values for the weights $[w(1) \ldots w(N)]$ at time $t+1$. If the weights are close to their ideal value (4.9), the value of $\hat{\mathbf{f}}_{t+1}$ obtained from the weighted decoding procedure will be of higher quality than that obtained from a more generic CS decoder. (Cossalter et al., 2010) explore two methods via which the weights at time $t+1$ can be selected using $\hat{\mathbf{f}}_t$ and $\hat{\mathbf{z}}_t$. The best of these consists of three steps: *1)* thresholding the entries of $\hat{\mathbf{f}}_t$, *2)* translating the thresholded silhouettes for a single time step according to the motion model and $\hat{\mathbf{z}}_t$, and *3)* dilating the translated silhouettes using a predefined dilation element. The final step accounts for uncertainty in the change of object appearance from one frame to the next. The result is a modified foreground image, which can then be interpreted as a prediction of $\mathbf{f}_{t+1}$. This prediction is used to define the weights according to (4.9), and the weighted decoding procedure is used to obtain $\hat{\mathbf{f}}_{t+1}$.

The above method is repeated at each new time instant. For a fixed compressive measurement rate, it is shown to provide more accurate foreground reconstructions than decoders that do not take advantage of the tracker output. Accordingly, it is also the case that such a method is able to more successfully tolerate lower bit rates. These results reveal the benefit of using the high level tracker information in compressive sensing systems.

4.4 Compressive sensing for multi-view tracking

Another direct application of CS to a data-rich tracking problem is presented by (Reddy et al., 2008). Specifically, a method for using multiple sensors to perform multi-view tracking employing a coding scheme based on compressive sensing is developed. Assuming that the observed data contains no background component (this could be realized, e.g., by preprocessing using any of the background subtraction techniques previously discussed), the method uses known information regarding the sensor geometry to facilitate a common data encoding scheme based on CS. After data from each camera is received at a central processing station, it is fused via CS decoding and the resulting image or three dimensional grid can be used for tracking.

The first case considered is one where all objects of interest exist in a known ground plane. It is assumed that the geometric transformation between it and each sensor plane is known. That is, if there are C cameras, then the *homographies* $\{\mathbf{H}_j\}_{j=1}^{C}$ are known. The relationship between coordinates (u, v) in the j^{th} image and the corresponding ground plane coordinates (x, y) is determined by $\mathbf{H}_j$ as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \mathbf{H}_j \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad , \tag{4.10}$$

where the coordinates are written in accordance with their homogeneous representation. Since $\mathbf{H}_j$ can vary widely across the set of cameras due to varying viewpoint, an encoding scheme designed to achieve a common data representation is presented. First, the ground plane is sampled, yielding a discrete set of coordinates $\{(x_i, y_i)\}_{i=1}^{N}$. An occupancy vector, $\mathbf{x}$, is defined over these coordinates, where $\mathbf{x}(n) = 1$ if foreground is present at the corresponding coordinates and is 0 otherwise. For each camera's observed foreground image in the set $\{\mathbf{I}_j\}_{j=1}^{C}$, an occupancy vector $\mathbf{y'}_j$ is formed as $\mathbf{y'}_j(i) = \mathbf{I}_j(u_i, v_i)$, where (u_i, v_i) are the (rounded) image plane coordinates corresponding to (x_i, y_i) obtained via (4.10). Thus, $\mathbf{y'}_j = \mathbf{x} + \mathbf{e}_j$, where $\mathbf{e}_j$ represents any error due to the coordinate rounding and other noise. Figure 4 illustrates the physical configuration of the system.

Noting that $\mathbf{x}$ is often sparse, the camera data $\{\mathbf{y'}_j\}_{j=1}^{C}$ is encoded using compressive sensing. First, C measurement matrices $\{\mathbf{\Phi}_j\}_{j=1}^{C}$ of equal dimension are formed according to a construction that affords them the RIP of appropriate order for $\mathbf{x}$. Next, the camera

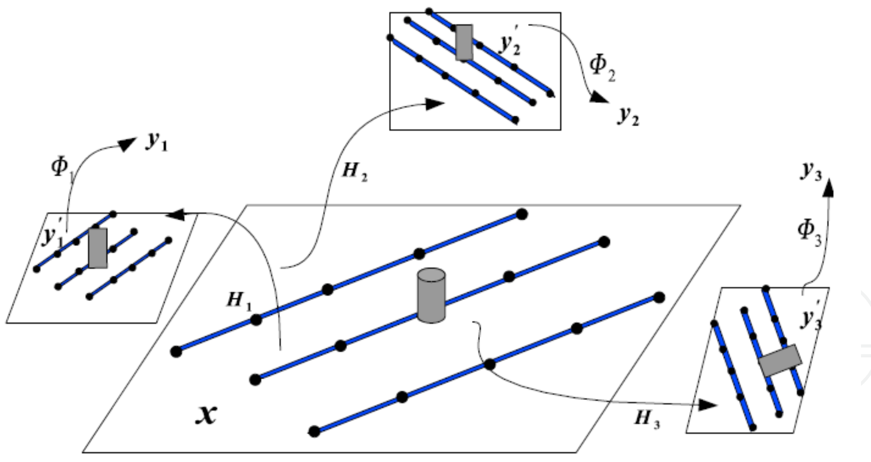

Fig. 4. Physical diagram capturing the assumed setup of the multi-view tracking scenario. Figure originally appears in (Reddy et al., 2008).

data is projected into the lower-dimensional space by computing $\mathbf{y}_j = \mathbf{\Phi}_j \mathbf{y}'_j$, $j = 1, \ldots, C$. This lower-dimensional data is transmitted to a central station, where it is ordered into the following structure:

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_C \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_1 \\ \vdots \\ \mathbf{\Phi}_C \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_C \end{bmatrix} \tag{4.11}$$

which can be written as $\mathbf{y} = \mathbf{\Phi x} + \mathbf{e}$. This is a noisy version of the standard CS problem presented in Section 3, and an estimate of $\mathbf{x}$ can be found using a relaxed version of (3.3), i.e.,

$$\hat{\mathbf{x}} = \min_{\mathbf{z} \in \mathbb{R}^N} \|\mathbf{z}\|_1 \quad \text{subject to} \quad \|\mathbf{\Phi z} - \mathbf{y}\|_2 \leq \|\mathbf{e}\|_2 \quad . \tag{4.12}$$

The estimated occupancy grid (formed, e.g., by thresholding $\hat{\mathbf{x}}$) can then be used as input to subsequent tracker components.

The above process is also extended to three dimensions, where $\mathbf{x}$ represents an occupancy grid over 3D space, and the geometric relationship in (4.10) is modified to account for the added dimension. The rest of the process is entirely similar to the two dimensional case. Of particular note is the advantage in computational complexity: it is only on the order of the dimension of $\mathbf{x}$ as opposed to the number of measurements received.

4.5 Compressive particle filtering

The final application of compressive sensing in tracking presented in this chapter is the compressive particle filtering algorithm developed by (Wang et al., 2009). As in Section 4.1, it is assumed that the system uses a sensor that is able to collect compressive measurements. The goal is to obtain tracks *without* having to perform CS decoding. That is, the method solves the sequential estimation problem using the compressive measurements directly, avoiding procedures such as (3.3). Specifically, the algorithm is a modification to the particle filter of Section 2.2.2.

First, the system is formulated in state space, where the state vector at time t is given by

$$\mathbf{s}_t = [s_t^x \; s_t^y \; \dot{s}_t^x \; \dot{s}_t^y \; \psi_t]^T \quad . \tag{4.13}$$

(s_t^x, s_t^y) and $(\dot{s}_t^x, \dot{s}_t^y)$ represent the object position and velocity in the image plane, and ψ_t is a parameter specifying the width of an appearance kernel. The appearance kernel is taken to be a Gaussian function defined over the image plane and centered at (s_t^x, s_t^y) with i.i.d. component variance proportional to ψ_t. That is, given $\mathbf{s}_t$, the j^{th} component of the vectorized image, $\mathbf{z}_t$, is defined as

$$\mathbf{z}_t^j(\mathbf{s}_t) = C_t \exp\{-\psi_t(\begin{bmatrix} s_k^x \\ s_k^y \end{bmatrix} - \mathbf{r}^j)\} \quad , \tag{4.14}$$

where $\mathbf{r}^j$ specifies the two dimensional coordinate vector belonging to the j^{th} component of $\mathbf{z}_t$.

The state equation is given by

$$\mathbf{s}_{t+1} = f_t(\mathbf{s}_t, \mathbf{v}_t) = \mathbf{D}\mathbf{s}_t + \mathbf{v}_t \quad , \tag{4.15}$$

where

$$D = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\alpha))$ for a preselected noise variance vector α.

The observation equation specifies the mapping from the state to the observed compressive measurements $\mathbf{y}_t$. If $\mathbf{\Phi}$ is the CS measurement matrix used to sense $\mathbf{z}_t$, this is given by

$$\mathbf{y}_t = \mathbf{\Phi}\mathbf{z}_t(\mathbf{s}_t) + \mathbf{w}_t \quad , \tag{4.17}$$

where $\mathbf{w}_t$ is zero-mean Gaussian measurement noise with covariance $\mathbf{\Sigma}$.

With the above specified, the bootstrap particle filtering algorithm presented in Section 2.2.2 can be used to sequentially estimate $\mathbf{s}_t$ from the observations $\mathbf{y}_t$. Specifically, the importance weights belonging to candidate samples $\{\tilde{\mathbf{s}}_t^{(i)}\}_{i=1}^N$ can be found via

$$\tilde{w}_t^{(i)} = p(\mathbf{y}_t | \tilde{\mathbf{s}}_t^{(i)}) = \mathcal{N}(\mathbf{y}_t; \mathbf{\Phi}\mathbf{z}_t(\tilde{\mathbf{s}}_t^{(i)}), \mathbf{\Sigma}) \tag{4.18}$$

and rescaling to normalize across all i. These importance weights can be calculated at each time step without having to perform CS decoding on $\mathbf{y}$. In some sense, the filter is acting purely on compressive measurements, and hence the name "compressive particle filter."

5. Summary

This chapter presented current applications of CS in visual tracking. In the presence of large quantities of data, algorithms common to classical tracking can become cumbersome. To provide context, a review of selected classical methods was given, including background subtraction, Kalman and particle filtering, and mean shift tracking. As a means by which data reduction can be accomplished, the emerging theory of compressive sensing was presented. Compressive sensing measurements $\mathbf{y} = \mathbf{\Phi x}$ necessitate a nonlinear decoding process, which makes accomplishing high-level tracking tasks difficult. Recent research addressing this problem was presented. Compressive background subtraction was discussed as a way to incorporate compressive sensors into a tracking system and obtain foreground-only images using a reduced amount of data. Kalman filtered CS was then discussed as a computationally and data-efficient way to track slowly moving objects. As an example of using high-level tracker information in a CS system, a method that uses it to improve the foreground estimate was presented. In the realm of multi-view tracking, CS was used as part of an encoding scheme that enabled computationally feasible occupancy map fusion in the presence of a large number of cameras. Finally, a compressive particle filtering method was discussed, via which tracks can be computed directly from compressive image measurements.

The above research represents significant progress in the field of performing high-level tasks such as tracking in the presence of data reduction schemes such like CS. However, there is certainly room for improvement. Just as CS was developed by considering the integration of sensing and compression, future research in this field must jointly consider sensing and the end-goal of the system, i.e., high-level information. Sensing strategies devised in accordance with such considerations should be able to efficiently handle the massive quantities of data present in modern surveillance systems by only sensing and processing that which will yield the most relevant information.

6. References

Anderson, B. & Moore, J. (1979). *Optimal Filtering*, Dover.

Baraniuk, R. (2011). More is less: signal processing and the data deluge., *Science* 331(6018): 717–9.

Baraniuk, R. G. (2007). Compressive Sensing [Lecture Notes], *IEEE Signal Processing Magazine* 24(4): 118–121.

Broida, T. & Chellappa, R. (1986). Estimation of object motion parameters from noisy images., *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(1): 90–9.

Bruckstein, A., Donoho, D. & Elad, M. (2009). From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images, *SIAM Review* 51(1): 34.

Candès, E. & Wakin, M. (2008). An introduction to compressive sampling, *IEEE Signal Processing Magazine* 25(2): 21–30.

Cevher, V., Sankaranarayanan, A., Duarte, M., Reddy, D., Baraniuk, R. & Chellappa, R. (2008). Compressive sensing for background subtraction, *ECCV 2008* .

Comaniciu, D. & Meer, P. (2002). Mean shift: a robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(5): 603–619.

Comaniciu, D., Ramesh, V. & Meer, P. (2003). Kernel-based object tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(5): 564–577.

Cossalter, M., Valenzise, G., Tagliasacchi, M. & Tubaro, S. (2010). Joint Compressive Video Coding and Analysis, *IEEE Transactions on Multimedia* 12(3): 168–183.

Doucet, A., de Freitas, N. & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*, Springer.

Duarte, M., Davenport, M., Takhar, D., Laska, J., Kelly, K. & Baraniuk, R. (2008). Single-Pixel Imaging via Compressive Sampling, *IEEE Signal Processing Magazine* 25(2): 83–91.

Elgammal, A., Duraiswami, R., Harwood, D. & Davis, L. (2002). Background and foreground modeling using nonparametric kernel density estimation for visual surveillance, *Proceedings of the IEEE* 90(7): 1151–1163.

Isard, M. & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density, *European Conference on Computer Vision* pp. 343–356.

Poor, H. V. (1994). *An Introduction to Signal Detection and Estimation, Second Edition*, Springer-Verlag.

Reddy, D., Sankaranarayanan, A., Cevher, V. & Chellappa, R. (2008). Compressed sensing for multi-view tracking and 3-D voxel reconstruction, *IEEE International Conference on Image Processing* (4): 221–224.

Romberg, J. (2008). Imaging via Compressive Sampling, *IEEE Signal Processing Magazine* 25(2): 14–20.

Sankaranarayanan, A. & Chellappa, R. (2008). Optimal Multi-View Fusion of Object Locations, *IEEE Workshop on Motion and Video Computing* pp. 1–8.

Sankaranarayanan, A., Veeraraghavan, A. & Chellappa, R. (2008). Object Detection, Tracking and Recognition for Multiple Smart Cameras, *Proceedings of the IEEE* 96(10): 1606–1624.

Stauffer, C. & Grimson, W. (1999). Adaptive background mixture models for real-time tracking, *IEEE Conference on Computer Vision and Pattern Recognition*.

Vaswani, N. (2008). Kalman filtered compressed sensing, *IEEE International Conference on Image Processing* (1): 893–896.

Wang, E., Silva, J. & Carin, L. (2009). Compressive particle filtering for target tracking, *IEEE Workshop on Statistical Signal Processing*, pp. 233–236.

Warnell, G., Reddy, D. & Chellappa, R. (2012). Adaptive Rate Compressive Sensing for Background Subtraction, *IEEE International Conference on Acoustics, Speech, and Signal Processing* .

Willett, R., Marcia, R. & Nichols, J. (2011). Compressed sensing for practical optical imaging systems: a tutorial, *Optical Engineering* 50(7).

Yilmaz, A., Javed, O. & Shah, M. (2006). Object Tracking: A Survey, *ACM Computing Surveys* 38(4).