# SPARSE REPRESENTATION FRAMEWORKS FOR INFERENCE PROBLEMS IN VISUAL SENSOR NETWORKS

by

Serhan Coşar

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Sabancı University

November, 2013

SPARSE REPRESENTATION FRAMEWORKS FOR INFERENCE PROBLEMS IN VISUAL SENSOR NETWORKS

APPROVED BY:

Assoc. Prof. Dr. Müjdat Çetin .....

(Dissertation Supervisor)

Assoc. Prof. Dr. Selim Balcısoy .....

Assist. Prof. Dr. Ali Özer Ercan .....

Prof. Dr. Aytül Erçil .....

Assoc. Prof. Dr. Hakan Erdoğan .....

DATE OF APPROVAL: 01.11.2013

SPARSE REPRESENTATION FRAMEWORKS FOR INFERENCE PROBLEMS IN VISUAL SENSOR NETWORKS

Serhan Coşar

Electronics Engineering, PhD Thesis, 2013

Thesis Supervisor: Assoc. Prof. Dr. Müjdat Çetin

Keywords: Visual sensor networks, camera networks, sparse representation, human tracking, compressing likelihoods functions, action recognition

## Abstract

Visual sensor networks (VSNs) form a new research area that merges computer vision and sensor networks. VSNs consist of small visual sensor nodes called camera nodes, which integrate an image sensor, an embedded processor, and a wireless transceiver. Having multiple cameras in a wireless network poses unique and challenging problems that do not exist either in computer vision or in sensor networks. Due to the resource constraints of the camera nodes, such as battery power and bandwidth, it is crucial to perform data processing and collaboration efficiently.

This thesis presents a number of sparse-representation based methods to be used in the context of surveillance tasks in VSNs. Performing surveillance tasks, such as tracking, recognition, etc., in a communication-constrained VSN environment is extremely challenging. Compressed sensing is a technique for acquiring and reconstructing a signal from small amount of measurements utilizing the prior knowledge that the signal has a sparse representation in a proper space. The ability of sparse representation tools to reconstruct signals from small amount of observations fits

well with the limitations in VSNs for processing, communication, and collaboration. Hence, this thesis presents novel sparsity-driven methods that can be used in action recognition and human tracking applications in VSNs.

A sparsity-driven action recognition method is proposed by casting the classification problem as an optimization problem. We solve the optimization problem by enforcing sparsity through $l_1$ regularization and perform action recognition. We have demonstrated the superiority of our method when observations are low-resolution, occluded, and noisy. To the best of our knowledge, this is the first action recognition method that uses sparse representation. In addition, we have proposed an adaptation of this method for VSN resource constraints. We have also performed an analysis of the role of sparsity in classification for two different action recognition problems.

We have proposed a feature compression framework for human tracking applications in visual sensor networks. In this framework, we perform decentralized tracking: each camera extracts useful features from the images it has observed and sends them to a fusion node which collects the multi-view image features and performs tracking. In tracking, extracting features usually results a likelihood function. To reduce communication in the network, we compress the likelihoods by first splitting them into blocks, and then transforming each block to a proper domain and taking only the most significant coefficients in this representation. To the best of our knowledge, compression of features computed in the context of tracking in a VSN has not been proposed in previous works. We have applied our method for indoor and outdoor tracking scenarios. Experimental results show that our approach can save up to 99.6% of the bandwidth compared to centralized approaches that compress raw images to

decrease the communication. We have also shown that our approach outperforms existing decentralized approaches.

Furthermore, we have extended this tracking framework and proposed a sparsity-driven approach for human tracking in VSNs. We have designed special overcomplete dictionaries that exploit the specific known geometry of the measurement scenario and used these dictionaries for sparse representation of likelihoods. By obtaining dictionaries that match the structure of the likelihood functions, we can represent likelihoods with few coefficients, and thereby decrease the communication in the network. This is the first method in the literature that uses sparse representation to compress likelihood functions and applies this idea for VSNs. We have tested our approach for indoor and outdoor tracking scenarios and demonstrated that our approach can achieve bandwidth reduction better than our feature compression framework. We have also presented that our approach outperforms existing decentralized and distributed approaches.

GÖRSEL ALGILAYICI AĞLARINDAKİ İSTATİSTİKSEL ÇIKARIM PROBLEMLERİ İÇİN SEYREK TEMSİL YÖNTEMLERİ

Serhan Coşar

Elektronik Mühendisliği, Doktora Tezi, 2013

Tez Danışmanı: Doç. Dr. Müjdat Çetin

Anahtar Sözcükler: Görsel algılayıcı ağları, kamera ağları, seyrek temsil, insan takibi, olabilirlik fonksiyonlarının sıkıştırılması, hareket tanıma

## Özet

Görsel algılayıcı ağları (GAAlar), görüntü işleme ve algılayıcı ağları konularını birleştiren yeni bir araştırma alanıdır. GAAlar, bir imge algılayıcı, bir gömülü işlemci ve bir kablosuz alıcı/vericiden oluşan kamera düğümleri denilen küçük görsel algılayıcı düğümlerinden oluşmaktadır. Bir kablosuz ağda birden fazla kameranın bulunması, görüntü işlemede ya da algılayıcı ağlarında olmayan, kendine has ve zor problemler yaratmaktadır. Kamera düğümlerindeki pil gücü ve bantgenişliği gibi kaynak kısıtları nedeniyle, veri işlemenin ve kameralar arasındaki işbirliğinin verimli bir şekilde yapılması çok önemlidir.

Bu tezde, GAAlarda gözetleme işlerinde kullanılmak üzere seyrek-temsil tabanlı yöntemler anlatılmaktadır. Haberleşmenin kısıtlı olduğu GAA ortamında, hedef takibi, tanıma, vb. gözetleme işleri yapmak son derece zordur. Sıkıştırılmış algılama, bir işaretin uygun bir uzayda seyrek temsili olduğu ön bilgisine kullanarak çok az sayıdaki gözlem verisinden işareti geri çatmak için kullanılan bir tekniktir. Seyrek temsil

araçlarının az sayıdaki gözlem verisinden işaretleri geri çatma özelliği, GAAlarda işleme, haberleşme ve işbirliği yaparken ortaya çıkan sınırlamalara çok uygundur. Dolayısıyla, bu tez GAAlardaki hareket tanıma ve insan takibi uygulamarında kullanılabilecek yeni seyreklik-güdümlü yöntemler sunmaktadır.

İlk olarak, sınıflandırma problemini bir optimizasyon problemine çeviren seyreklik-güdümlü bir hareket tanıma yöntemi önerilmiştir. Optimizasyon problemi $l_1$ düzenleştiricisi ile seyreklik zorlayarak çözülmekte ve hareket tanıma gerçekleştirilmektedir. Yöntemimizin üstünlüğü gözlem verilerinin düşük-çözünürlükte, engellenmiş ve gürültülü olduğu durumlarda gösterilmiştir. Bildiğimiz kadarıyla, bu yöntem seyrek temsil kullanan ilk hareket tanıma yöntemidir. Ek olarak, bu yöntem GAA kaynak kısıtlarına uygun bir hale de dönüştürülmüştür. Ayrıca, iki farklı hareket tanıma problemi kullanılarak seyrekliğin sınıflandırmadaki etkisi incelenmiştir.

İkinci olarak, görsel algılayıcı ağlarındaki insan takibi uygulamaları için bir öznitelik sıkıştırma yöntemi önerilmiştir. Bu yöntemde, merkezi olmayan takip işlemi gerçekleştirilmiştir: her kamera kendi elde ettiği imgelerden önemli öznitelikleri çıkartır ve onları, çok-görüşlü imge özniteliklerini toplayan ve takibi gerçekleştiren bir füzyon düğümüne gönderir. Takip işleminde, öznitelik çıkartmak genelde bir olabilirlik fonksiyonu yaratmaktadır. Ağdaki haberleşmeyi azaltmak için, bu fonksiyonlar, önce bloklara ayırılarak, her blok uygun bir uzaya dönüştürülerek ve bu temsildeki sadece en önemli katsayıları alınarak, sıkıştırılmaktadır. Bildiğimiz kadarıyla, GAAlardaki takip uygulamalarında elde edilen özniteliklerin sıkıştırılması daha önce hiç önerilmemiştir. Yöntemimiz, iç mekan ve dış mekan takip senaryolarında uygulanmıştır. Deneysel sonuçlar göstermektedir ki, yöntemimiz haberleşmeyi azaltmak

için imgeleri sıkıştıran merkezi yöntemlere kıyasla bantgenişliğinin %99.6'sını kazandırabilmektedir. Ayrıca, yöntemimizin varolan merkezi olmayan yöntemlerden daha iyi çalıştığı da gösterilmiştir.

Son olarak, yukarıdaki takip yöntemi geliştirilmiş ve GAAlarda insan takibi için seyreklilik-güdümlü bir yöntem önerilmiştir. Gözlem senaryosundaki belirgin geometriden yararlanan özel sözlükler tasarlanmış ve bu sözlükler olabilirlik fonksiyonlarının seyrek temsilinde kullanılmıştır. Olabilirlik fonksiyonlarının yapısına uyumlu sözlükler elde ederek, fonksiyonlar çok az sayıda katsayı ile temsil edebilmekte, böylece ağdaki haberleşme azaltılabilmektedir. Bu yöntem seyrek temsil kullanarak olabilirlik fonksiyonlarını sıkıştıran ve GAAlarda bu fikri uygulayan literatürdeki ilk yöntemdir. Yöntemimiz iç mekan ve dış mekan takip senaryolarında test edilmiştir ve yöntemimizin öznitelik sıkıştırma yöntemimizden daha çok bantgenişliği kazandırdığı gösterilmiştir. Ayrıca, yöntemimizin var olan merkezi olmayan ve dağıtık yöntemlerden da daha iyi çalıştığı gösterilmiştir.

**Acknowledgements**

I take the pleasure of thanking everyone who made this thesis possible. Foremost, I owe my deepest gratitude to Assoc. Prof. Dr. Müjdat Çetin who has guided me throughout this thesis with his valuable suggestions and advices, most importantly for giving me the opportunity to work on my own research directions. I had an opportunity to learn from his experience and vision while working closely with him. I could not have imagined a better advisor and mentor for my PhD. I also greatly appreciate his dedication in reading and correcting this manuscript.

I am thankful to all my thesis committee members Assoc. Prof. Dr. Hakan Erdoğan, Prof. Dr. Aytül Erçil, Assoc. Prof. Dr. Selim Balcısoy and Assist. Prof. Dr. Ali Özer Ercan for the time spent in reading this manuscript and for their valuable suggestions in improving the final version of the thesis. My special thanks to Assoc. Prof. Dr. Hakan Erdoğan and Prof. Dr. Mustafa Ünel for their valuable time and input throughout this thesis. The work presented here has been funded by the graduate scholarship of the Scientific and Technological Research Council of Turkey. I am grateful for their support.

I thank all the valuable members of VPA laboratory: for creating such a friendly working environment and for the many great moments spent together. I thank Özge Batu, İ. Saygın Topkaya, N. Özben Önhon, Emad Mounir Grais for the exciting, funny and fruitful discussions we had together. I also thank our VPA admin, Osman Rahmi Fıçıcı, for always being kind and providing the resources required for this thesis.

I thank my family members, my mother Aysun and my father Orhan for their love, unconditional support and guidance in all my life. My special thanks to my mother-in-law Aysel, my father-in-law İlami, and my sister-in-law Eylem for their never-ending support. Their support and love have helped me through the most difficult periods of this PhD study.

Huge thanks to my wife, Pınar, for her patience, endless love and support towards me during the hardest years of my PhD studies. Being with a PhD student is not an easy task. She was right beside me and she has raised my confidence whenever necessary and she has always believed in me more than myself.

# TABLE OF CONTENTS

# List of Figures

# List of Tables

# 1 INTRODUCTION

The word *system* has a long history which can be traced back to Plato, Aristotle, and Euclid. It had meant "total", "crowd", or "union". In modern times, in natural sciences and information technology, we basically define a system as a set of components forming an integrated whole that has inputs, outputs, and a processor. As in the first use of the term in natural sciences by the French physicist Nicolas Lonard Sadi Carnot in 19th century for thermodynamic systems, a system is built by integrating the components physically. With the advances in telecommunication technology, physical integration of components become unnecessary. Now, we are in an era in which the robotic rover *Curiosity* in Mars, that takes commands (inputs) from Earth and sends its observations (outputs) back, can be defined as a system.

This telecommunication breakthrough together with the advances in microelectromechanical technology has resulted in the production of autonomous systems that can monitor physical or environmental conditions, such as temperature, sound, pressure, etc., cooperatively process the data and transmit extracted information to remote locations. These systems are called wireless sensor networks. More recently, the availability of inexpensive hardware such as CMOS cameras that are able to capture visual data from the environment has supported the development of Visual Sensor Networks (VSNs), i.e., networks of wirelessly interconnected devices that acquire

video data. This new technology provides a number of potential applications, ranging from security to monitoring, and 3D modeling to telepresence.

In the past two decades, the increase of theft, organized crime and terrorist attacks has created the need for new security systems that use surveillance cameras. A huge number of cameras have been installed, and are still being installed, in the streets of big cities. Even in a small grocery store, there are a number of security cameras to prevent theft. Mostly, the cameras are installed on networks in which surveillance cameras act as independent peers that continuously send video streams to a central processing server, where the video is analyzed by a human operator and recorded. But, as the number of cameras increases video analysis becomes a challenge. Although this vast number of cameras are usually installed in wired networks, with the availability of tiny small smart cameras such as Google Glass, the community will switch to VSNs and we will benefit from lots of potential applications that will come in to the stage with VSNs. For the wired camera networks, maintenance is very hard and the non-flexible system does not allow changing the locations of the cameras. Together with wireless mobility, VSNs bring out new security applications that are portable and easily deployable. For instance, it is possible to setup flexible and mobile autonomous surveillance systems for monitoring and securing concerts, demonstrations, etc. Such systems can also provide an important support for special operations of police forces and surveillance of borders. An important application of VSNs is swarm robotics. In applications that involve multiple mobile robots, such as humanoids, unmanned aerial vehicles (UAVs) and autonomous underwater vehicles (AUVs), it is crucial to perform tasks in a collaborative manner. Since a network of mobile robots resembles visual sensor networks, the technologies developed for VSNs

can potentially be used for swarm robotics.

Visual sensor networks are capable of local image processing and data communication. Different from traditional camera-based surveillance networks where cameras simply stream their image data to a centralized server for processing, cameras in VSNs form a distributed system, performing information extraction and collaborating on application-specific tasks. Due to the resource constraints of the camera nodes such as restricted computation capacity, remaining battery power, and available bandwidth, VSNs usually need to limit the amount of data being exchanged among the camera nodes and the complexity of algorithms running on the camera nodes. Thus, performing surveillance tasks, such as tracking, recognition, etc., in a communication-constrained VSN environment is extremely challenging.

Parsimony has a rich history as a guiding principle for inference [11]. One of its most celebrated examples, the principle of minimum description length in model selection, enables choosing a limited subset of features or models from the training data, rather than directly using the data for representing or classifying an input signal. In statistical signal processing, over the last ten years, the idea of searching for parsimony led to an alternative sampling/sensing theory, called "compressed sensing" and "sparse representation", makes it possible to recover signals, images, and other data from small amount of observations. This thesis aims at developing algorithms for action recognition and human tracking in VSNs by using the ideas of compressed sensing and sparse representation.

## 1.1 Problem Definition and State-of-the-art

Using a camera in a wireless network leads to unique and challenging problems that are more complex than the traditional multi-camera video analysis systems and wireless sensor networks might have. In wireless sensor networks, most sensors provide measurements of temporal signals that represent physical quantities such as temperature. On the other hand, in VSNs, at each time instant image sensors provide a 2D set of data points, which we see as an image. This richer information content increases the complexity of data processing and analysis. The requirements of resources such as energy and bandwidth forms one of the main problems.

In this thesis, we focus on problems caused by the communication constraint for action recognition and human tracking applications in VSNs. In most of the multi-camera systems, a centralized approach, in which the raw data acquired by cameras are compressed, collected in a central unit and analyzed to perform the task of interest, is followed. With a data compression perspective, the common approach is to compress images in the process of transmitting them to the central unit. In this strategy, the main aim is on low-level communication. The communication load is decreased by compressing the raw data without considering the final inference goal based on the information content of the data. Compressing images will affect the quality of the images and using compressed images may decrease the performance of further inference tasks. Thus, for performing complex tasks, such as tracking, recognition, etc., this strategy is not appropriate.

To minimize the amount of data to be communicated, in some methods simple fea-

tures are used for communication. For instance, 2D trajectories are used in [12]. In [13], 3D trajectories together with color histograms are used. Hue histograms along with 2D position are used in [14]. Moreover, there are decentralized approaches in which cameras are grouped into clusters and tracking is performed by local cluster fusion nodes using the features extracted by the cameras in the cluster. This kind of approaches have been applied to the multi-camera target tracking problem in various ways [15, 7, 16]. To further increase scalability and to reduce communication costs, there are distributed methods that perform processing and analysis in all cameras in a distributed fashion, without local fusion centers. Each camera performs estimation locally and transmits to its neighbors. The received estimates are used to refine the next-camera estimates, and these refined estimates are then transmitted to the next neighbor [9, 17].

Previous works proposed for VSNs have some handicaps. The methods in [12, 13, 14] that use simpler features may be capable of decreasing the communication, but they are not capable of maintaining robustness. In order to adapt to bandwidth constraints, these methods choose to change the features from complex and robust to simpler but not so effective ones. For instance, since color information extracted by each camera depends on the lightning conditions and image sensor, there may be variability for the color of the same person at each camera view. Thus, color histograms are not robust features. Using color histograms may fail association of the information coming from cameras. As in [15, 7, 16], performing local processing and collecting features to the fusion node may not satisfy the bandwidth requirements in a communication-constrained VSN environment. In particular, depending on the size of image features and the number of cameras in the network, even collecting

features to the fusion node may become expensive for the network. In such cases, further approximations on features are necessary. An efficient approach that reduces the bandwidth requirements without significantly decreasing the quality of image features is needed. A disadvantage of distributed methods in [9, 17] is that the inference is drawn in a distributed fashion. In order to use such algorithms in a VSN environment, we need to implement existing centralized trackers in a distributed way. To do that, we have to change each step from feature extraction to final inference, which is not a straight-forward task and which can affect the performance of the tracker.

In this thesis, we propose different strategies that are better matched to the final inference goals, that, in the context of this thesis, are action recognition and tracking.

## 1.2 Contributions

The main goal of this thesis is to propose novel sparsity-driven methods to solve communication constraint problems for action recognition and human tracking systems in VSNs. We have made three distinct contributions:

- A sparse representation based action recognition method
- A feature compression framework for human tracking
- A sparse representation framework for human tracking

For the action recognition problem, based on the assumption that a test sample can be written as a linear combination of training samples from the class it belongs, we

cast the classification problem as an optimization problem and solve it by enforcing sparsity through $l_1$ regularization. In addition, we have proposed an approach to adapt this method for VSN resource constraints. In recent studies, the role of sparsity in classification has been questioned and it has been argued that $l_1$-norm constraint may not be necessary [18, 19]. Following these studies, we have also analyzed the role of sparsity in classification for two different action recognition problems.

A feature compression framework is proposed to overcome communication problems of human tracking systems in visual sensor networks. In this framework, tracking is performed in a decentralized way: each camera extracts useful features from the images it has observed and sends them to a fusion node which collects the multi-view image features and performs tracking. In tracking, extracting features usually results in a likelihood function. Instead of sending the likelihood functions themselves to the fusion node, we compress the likelihoods by first splitting them into blocks, and then transforming each block to a proper domain and taking only the most significant coefficients in this representation. By sending the most significant coefficients to the fusion node, we decrease the communication in the network.

As an extension of this framework, we have proposed a sparsity-driven approach for human tracking applications. We have designed special overcomplete dictionaries that are matched to the structure of the likelihood functions and used these dictionaries for sparse representation of likelihoods. In particular our dictionaries are designed by exploiting the specific known geometry of the measurement scenario and by focusing on the problem of human tracking. Each element in the dictionary for each camera corresponds to the likelihood that would result from a single human at a

particular location in the scene. Hence, by using these dictionaries, we can represent likelihoods with few coefficients, and thereby decrease the communication between cameras and fusion nodes.

The tracking frameworks fit well to the needs of the VSN environment in two aspects: i) the processing capabilities of cameras in the network are utilized by extracting image features at the camera-level, ii) using only the most significant coefficients, obtained either from block-based compression scheme or sparse representation of likelihoods, in network communication saves energy and bandwidth resources. We have achieved a goal-directed compression scheme for the tracking problem in VSNs by performing local processing at the nodes and compressing the resulting likelihood functions which are related to the tracking goal, rather than compressing raw images.

Another advantage of these frameworks is that they are generic frameworks that do not require the use of a specific tracking method. Usually in tracking, a likelihood function is obtained in order to perform estimation. Thus, our frameworks can work together with any kind of probabilistic tracking algorithm. Without making significant changes on existing tracking methods, which may degrade the performance, existing methods can be used within our frameworks in VSN environments.

## 1.3 Outline

In Chapter 2, we provide some technical background about topics on which the thesis is based. Chapter 3 presents our novel multi-camera action recognition method that is based on sparse representation. A feature compression framework that is applied for

likelihood functions is described in Chapter 4. In Chapter 5, we extend this feature compression framework and describe our sparse representation based framework for tracking. Finally, in Chapter 6 we draw conclusions and talk about ideas for future directions.

# 2 BACKGROUND

In this chapter, topics that provide a basis for this thesis are reviewed. In the first section, the theory of compressed sensing and sparse representation is described and some examples for applications on computer vision problems are discussed. In the second section, the new field of visual sensor networks is introduced and the problems of setting up a surveillance system in visual sensor networks are described in detail.

## 2.1 Sparse Representation and Compressed Sensing

### 2.1.1 Overview

The Shannon/Nyquist sampling theory, which states that the number of samples required to capture a signal must be at least twice the maximum frequency present in the signal, is one of the central principles of signal processing. However, it is well known that the Nyquist rate is a sufficient, but not a necessary, condition. Over the last ten years, an alternative sampling/sensing theory, known as "compressed sensing", has been proposed to recover signals, images, and other data from what appear to be undersampled observations [20].

Two important observations are at the heart of this new approach. The first is that the Shannon/Nyquist signal representation uses only minimal prior knowledge about the signal being sampled, i.e. its bandwidth. However, most signals we are interested in are structured and depend upon a smaller number of degrees of freedom than the bandwidth suggests. In other words, most signals of interest are *sparse* or compressible in the sense that they can be encoded with just a few numbers without much numerical or perceptual loss. The second observation is that the useful information in compressible signals can be captured via sampling or sensing protocols that directly condense signals into a small amount of data. A surprise is that many such sensing protocols do nothing more than linearly correlate the signal with a fixed set of signal-independent waveforms. These waveforms, however, need to be "incoherent" with the family of waveforms in which the signal is compressible. One then typically uses numerical optimization to reconstruct the signal from the linear measurements.

Although parsimony is a long lasting phenomenon in philosophy, it also has a rich history in statistical inference. It is used as a principle for choosing a limited subset of features or models from the training data, rather than directly using the data for representing or classifying an input signal [11]. In the statistical signal processing community, starting from the linear transforms, such as Fourier, DCT, to non-linear transfoms such as, STFT, Gabor, Wavelets, we have searched for *compaction* which will later be replaced with *sparsity*. More recently, searching for parsimony corresponds to the algorithmic problem of computing sparse linear representations with respect to an overcomplete dictionary of base elements or signal atoms. Since, it is known that natural images can be sparsely represented in wavelet domain [21], *sparse representation* (SR) of signals became a very popular field.

As it is expressed above, compressed sensing (CS) is a technique for acquiring and reconstructing a signal from small amount of measurements utilizing the prior knowledge that the signal has a sparse representation in a proper space. To make this possible, CS relies on two principles: sparsity, which is related to the signals of interest, and incoherence, which is about the sensing modality. Sparsity expresses the idea that the "information rate" of a continuous time signal may be much smaller than suggested by its bandwidth, or that a discrete-time signal depends on a number of degrees of freedom which is comparably much smaller than its (finite) length. More precisely, CS exploits the fact that many natural signals are sparse or compressible in the sense that they have concise representations when expressed in the proper basis. Incoherence extends the duality between time and frequency and expresses the idea that signals having a sparse representation in the proper basis must be spread out in the domain in which they are acquired, just as a Dirac or a spike in the time domain is spread out in the frequency domain. In other words, incoherence says that unlike the signal of interest, the sampling/sensing waveforms have an extremely dense representation in the proper basis. Further, there is a way to use numerical optimization to reconstruct the full-length signal from the small amount of collected data.

Following the technical aspects above, CS and SR have become important signal recovery techniques because of their success for acquiring, representing, and compressing high-dimensional signals in various application areas [2, 22, 23, 24]. This success is mainly due to the fact that important classes of signals such as audio and images have naturally sparse representations with respect to fixed bases (i.e., Fourier, wavelet). In addition, efficient and effective algorithms based on convex optimization

or greedy pursuit are available for computing such representations [25]. In the past few years, variations and extensions of $l_1$ minimization have been applied to many vision tasks, including face recognition [11], denoising and inpainting [22], background modeling [26], and image classification [27]. In almost all of these applications, using sparsity as a prior leads to state-of-the-art results [3].

Consider a signal $f$ which is obtained by linear functionals recording the values:

$$b_k = \langle f, \phi_k \rangle, \ k = 1, ..., m. \tag{2.1}$$

That is, we simply correlate the object we wish to acquire with the waveforms $\phi_k$. This is a standard setup. If the sensing waveforms are Dirac delta functions, for example, then $b$ is a vector of sampled values of $f$ in the time or space domain. If the sensing waveforms are sinusoids, then $b$ is a vector of Fourier coefficients. We are interested in *undersampled* situations in which the number $m$ of available measurements is much smaller than the dimension ($n$) of the signal $f$. For a variety of reasons, this is a common case in most of the problems. For instance, the number of sensors may be limited or the measurements may be extremely expensive as in certain imaging processes. As one would need to solve an underdetermined linear system of equations, recovering the signal $f$ looks rather a difficult task. Letting $\Phi$ denote the $m \times n$ sensing matrix with the vectors $\phi_1, \cdots, \phi_m$ as rows, the process of recovering $f \in \mathrm{R}^n$ from $b = \Phi f \in \mathrm{R}^m$ is ill-posed in general when $m < n$: there are infinitely many candidate signals $\tilde{f}$ for which $\Phi \tilde{f} = b$. But one could perhaps imagine a way out by relying on realistic models of signals $f$ which naturally exist.

As we have expressed previously, many natural signals have concise representations

when expressed in a convenient basis. Mathematically, we have a vector $f \in \mathrm{R}^n$ which we expand in an orthonormal basis $\Psi = [\psi_1 \psi_2 ... \psi_n]$ as follows:

$$f = \sum_{i=1}^{n} x_i \psi_i, \tag{2.2}$$

where $x$ is the coefficient vector of $f$, $x_i = \langle f, \psi_i \rangle$. It will be convenient to express $f$ as $\Psi x$ (where $\Psi$ is the $n \times n$ matrix with $\psi_1$ $\psi_2 ... \psi_n$ as columns). If many of the values in the coefficient vector, $x_i$, are close to or equal to zero, this implies the sparsity of $f$. When a signal has a sparse expansion, one can discard the small coefficients without much perceptual loss. Formally, consider $f_S$ obtained by keeping only the terms corresponding to the $S$ largest values of $(x_i)$ in the expansion in Eq. 2.2. By definition, $f_S = \Psi x_S$, where $x_S$ is the vector of coefficients $(x_i)$ with all but the largest $S$ elements set to zero. This vector is sparse in a strict sense since all but a few of its entries are zero; we will call *$S$–sparse* to such signals with at most $S$ nonzero entries. Since $\Psi$ is an orthonormal basis, we have $||f - f_S||_2 = ||x - x_S||_2$, and if $x$ is sparse or compressible in the sense that the sorted magnitudes of the $(x_i)$ decay quickly, then $x$ is well approximated by $x_S$ and, therefore, the error $||f - f_S||_2$ is small. In simple terms, one can "throw away" a large fraction of the coefficients without much loss. This principle is, of course, what underlies most modern lossy coders such as JPEG-2000 [28] and many others.

Ideally, we would like to measure all the $n$ coefficients of $f$, but we only observe a subset of these and collect the data

$$b_k = \langle f, \phi_k \rangle, \qquad k \in \{1, ..., m\}, \; m < n \tag{2.3}$$

With this information, we can recover the signal by first finding the sparse representation $x$ and then multiplying it with the basis functions, $f = \Psi x$. By putting $l_0$-norm, which counts the number of nonzero entries in a vector, constraint on $x$ we can find the sparse representation:

$$\min_{x \in \mathrm{R}^n} ||x||_0 \quad subject\ to\ \ b_k = \langle \phi_k, \Psi x \rangle, \ \ \forall k \in \{1, \cdots, m\} \tag{2.4}$$

However, the problem of finding the sparsest solution of an underdetermined system of linear equations is NP-hard. But, following the theories of CS and SR, it is shown that if the solution $x$ is *sparse enough*, the solution of the $l_0$-minimization problem in Eq. 2.4 is equal to the solution to the following $l_1$-minimization problem [29] ( $||x||_1 = \sum_i |x_i|$ ):

$$\min_{x \in \mathrm{R}^n} ||x||_1 \quad subject\ to\ \ b_k = \langle \phi_k, \Psi x \rangle, \ \ \forall k \in \{1, \cdots, m\} \tag{2.5}$$

That is, among all signals $f^* = \Psi x^*$ consistent with the data, where $x^*$ is the solution of the problem above, we pick that whose coefficient sequence has minimal $l_1$ norm.

When we have noisy observations, we have the linear system of $b_k = \langle f, \phi_k \rangle + e$, where $e$ is an unknown error term. In such a case, the optimization problem in Eq. 2.5 becomes as follows:

$$\min_{x \in \mathrm{R}^n} ||x||_1 \quad subject\ to\ \ ||b - Ax||_2 \leq \epsilon, \tag{2.6}$$

where $\epsilon$ bounds the amount of noise in data and the matrix $A = \Phi\Psi \in \mathrm{R}^{m \times n}$ is an overcomplete dictionary matrix, i.e. a matrix which has more columns than its rows.

The term of *sparse enough* above is vague. In [30], suppose the coefficients $x$ of

$f \in \mathrm{R}^n$ in the basis $\Psi$ is $S - sparse$, it is shown that if we select

$$m \geq C \cdot \mu^2(\Phi, \Psi) \cdot S \cdot \log n \tag{2.7}$$

measurements in random, the solution to Eq. 2.5 is exact to the solution to Eq. 2.4. $\mu(\Phi, \Psi)$ represents the coherence between the sending basis $\Phi$ and the representation basis $\Psi$ and defined by measuring the largest correlation between any two elements of $\Phi$ and $\Psi$ as

$$\mu(\Phi, \Psi) = \sqrt{n} \cdot \max_{1 \leq k,j \leq n} |\langle \psi_k, \psi_j \rangle| \tag{2.8}$$

This implies the relation between incoherence and the number of measurements required: the smaller the coherence, the fewer samples are needed.

After seeing the theoretical aspects of CS and SR, one may ask about how to solve the optimization problem in Eq. 2.5. The algorithms to solve such problems are explained in Section 2.1.2. While forming the matrix $A$, we may need to go further than wavelet transforms provide us. In Section 2.1.3, we have discussed about learning strategies to form the matrix $A$ that fits our requirements. Some examples on how the CS and SR phenomenons are applied on computer vision problems are described in Section 2.1.4.

### 2.1.2 Algorithms

In this section, we step to the practical aspects of CS and SR and go through the well-known algorithms that are used to solve $l_1$-minimization problems (For more details, please refer to [31]). The performance analysis for these algorithms can be found in experiments in Section 5.2.

First, we are going to discuss a classical solution to the $l_1$-min problem in Eq. 2.5, called the primal-dual interior-point method. A compact version of the problem in Eq. 2.5 is as follows:

$$\min_{x \in \mathrm{R}^n} ||x||_1 \quad subject\ to\ \ b = Ax \tag{2.9}$$

For the sake of simplicity, assuming that the sparse solution $x$ is nonnegative. Under this assumption, it is easy to see that the problem can be converted to the standard primal and dual forms in linear programming:

$$\begin{array}{cc} \textbf{Primal (P)} & \textbf{Dual (D)} \\ \min_x c^T x & \max_{y,z} b^T y \\ subj.\ to\ \ Ax = b & subj.\ to\ \ A^T y + z = c \\ x \geq 0 & z \geq 0 \end{array} \tag{2.10}$$

where for $l_1$-minimization, $c = \vec{1}$. The primal-dual algorithm simultaneously solves for the primal and dual optimal solutions [32]. It was proposed in [33] that (P) can be converted to a family of logarithmic barrier problems:

$$(P_\mu): \quad \begin{array}{c} \min_x c^T x - \mu \sum_{i=1}^n \log x_i \\ subj.\ to\ \ Ax = b, x > 0 \end{array} \tag{2.11}$$

Clearly, a feasible solution $x$ to ($P_\mu$ ) cannot have zero coefficients. Therefore, we define the interiors of the solution domains for (P) and (D) as:

$$P_{++} = \quad \{x : Ax = b, x > 0\}, \tag{2.12}$$

$$D_{++} = \quad \{(y, z) : A^T y + z = c, z > 0\},$$

$$S_{++} = \quad P_{++} \times D_{++} \tag{2.13}$$

Assuming that the above sets are non-empty, it can be shown that ($P_\mu$) has a unique global optimal solution $x(\mu)$ for all $\mu > 0$. As $\mu \to 0$, $x(\mu)$ and $(y(\mu), z(\mu))$ converge

to optimal solutions of problems (P) and (D) respectively [34, 35].

The primal-dual interior-point algorithm seeks the domain of the central trajectory for the problems (P) and (D) in $S_{++}$, where the central trajectory is defined as the set $S = (x(\mu), y(\mu), z(\mu)) : \mu > 0$ of solutions to the following system of equations:

$$\begin{aligned} XZ\vec{1} = \mu\vec{1}, \;\; Ax = y, \;\; A^T w + z = c, \\ x \geq 0, z \geq 0. \end{aligned} \tag{2.14}$$

where $X$ and $Z$ are square matrices with the coefficients of $x$ and $z$ as its diagonal, respectively, and zero otherwise ( e.g. $X = diag(x_1, x_2, \cdots, x_n) \in \mathrm{R}^{n \times n}$ ). The above condition is also known as the Karush-Kuhn-Tucker (KKT) conditions for the convex program $(P_\mu)$ [35, 36]. Hence, the update rule on the current value $(x^{(k)}, y^{(k)}, z^{(k)})$ is defined by the Newton direction $(\Delta x, \Delta y, \Delta z)$, which is computed as the solution to the following set of linear equations:

$$\begin{aligned} Z^{(k)}\Delta x + X^{(k)}\Delta z &= \hat{\mu}1 - X^{(k)}z^{(k)}, \\ A\Delta x &= 0, \\ A^T\Delta y + \Delta z &= 0 \end{aligned} \tag{2.15}$$

where $\hat{\mu}$ is a penalty parameter that is generally different from $\mu$ in $(P_\mu)$.

Algorithm 1 summarizes a conceptual implementation of the interior-point methods [1].

[1] The CVX library at `http://cvxr.com/cvx/` is a primal-dual interior-point solver implemented in MATLAB.

**Algorithm 2.1:** Primal-Dual Interior-Point Algorithm (PDIPA)

**input** : A full rank matrix $A \in \mathrm{R}^{m \times n}$ , $m < n$, a vector $b \in \mathrm{R}^m$ , initialization $(x^{(0)}, y^{(0)}, z^{(0)})$. Iteration $k \leftarrow 0$. Initial penalty $\mu$ and a decreasing factor $0 < \delta < n$.

**repeat**

$k \leftarrow k + 1$, $\mu \leftarrow \mu(1 - \delta/n)$ ;

Solve Eq. 2.15 for $(\Delta x, \Delta y, \Delta z)$ ;

$x^{(k)} \leftarrow x^{(k-1)} + \Delta x$ , $y^{(k)} \leftarrow y^{(k-1)} + \Delta y$ , $z^{(k)} \leftarrow z^{(k-1)} + \Delta z$ ;

**until** *stopping criteria is satisfied.*;

**output**: $x^* \leftarrow x^{(k)}$

### Gradient Projection Methods

Gradient projection (GP) methods try to find a sparse representation $x$ along a certain gradient direction, which induces much faster convergence speed. In this approach, the $l_1$-min problem (Eq. 2.6) is reformulated as a quadratic programming (QP) problem.

The $l_1$-min problem is equivalent to the so-called LASSO problem [37]:

$$\min_x ||b - Ax||_2^2 \ \ subject\ to\ ||x||_1 \leq \sigma, \tag{2.16}$$

where $\sigma > 0$ is an appropriately chosen constant. Using the Lagrangian method, the problem can be rewritten as an unconstrained problem:

$$x^* = \arg\min_{\infty} \frac{1}{2}||b - Ax||_2^2 + \lambda||x||_1, \tag{2.17}$$

where $\lambda$ is the Lagrangian multiplier.

By using truncated Newton interior-point method (TNIPM) [38] [2], the problem in

[2] A MATLAB Toolbox for TNIPM called L1LS is available at `http://www.stanford.edu/~boyd/l1_ls/`.

Eq. 2.17 is transformed to a quadratic problem but with inequality constraints:

$$\min \frac{1}{2}||Ax-b||_2^2 + \lambda \sum_{i=1}^{n} u_i$$

$$subject\ to\ -u_i \leq x_i \leq u_i,\ i = 1, \cdots, n \tag{2.18}$$

Then a logarithmic barrier for the constraints $-u_i \leq x_i \leq u_i$ can be constructed [33]:

$$\Phi(x,u) = -\sum_i \log(u_i + x_i) - \sum_i \log(u_i - x_i) \tag{2.19}$$

Over the domain of $(x,u)$, the central path consists of the unique minimizer $(x^*(t), u^*(t))$ of the convex function

$$F_t(x,u) = t(||Ax-b||_2^2 + \lambda \sum_{i=1}^{n} u_i) + \Phi(x,u), \tag{2.20}$$

where the parameter $t \in [0, \infty)$.

Using PDIPA, explained above, the optimal search direction using Newton's method is computed by

$$\nabla^2 F_t(x,u) \cdot \begin{bmatrix} \nabla x \\ \nabla u \end{bmatrix} = -\nabla F_t(x,u) \in \mathrm{R}^{2n} \tag{2.21}$$

For large-scale problems, directly solving Eq. 2.21 is computationally expensive. In [38], the search step is accelerated by a preconditioned conjugate gradients (PCG) algorithm, where an efficient preconditioner is proposed to approximate the Hessian of $\frac{1}{2}||Ax - |b|_2^2$.

### Homotopy Methods

One of the drawbacks of the PDIPA method is that they require the solution sequence $x(\mu)$ to be close to a "central path" as $\mu \to 0$, which sometimes is difficult to satisfy and computationally expensive in practice. There is an approach called Homotopy methods [39, 40] that can lessen these issues.

We recall that Eq. 2.6 can be written as an unconstrained convex optimization problem:

$$x^* = \arg\min_{\infty} F(x) = \arg\min_{\infty} f(x) + \lambda g(x), \tag{2.22}$$

where $f(x) = \frac{1}{2}||b - Ax||_2^2$ , $g(x) = ||x||_1$ , and $\lambda > 0$ is the Lagrange multiplier. On one hand, with respect to a fixed $\lambda$, the optimal solution is achieved when $0 \in \partial F(x)$. On the other hand, similar to the interior-point algorithm, if we define

$$\chi = \{x_\lambda^* : \lambda \in [0, \infty)\} \tag{2.23}$$

$\chi$ identifies a solution path that follows the change in $\lambda$: when $\lambda \to \infty$, $x_\lambda^* = 0$; when $\lambda \to 0$, $x_\lambda^*$ converges to the solution of Eq. 2.9.

The Homotopy methods exploit the fact that the objective function $F(x)$ undergoes a homotopy from the $l_2$ constraint to the $l_1$ objective in Eq. 2.22 as $\lambda$ decreases. One can further show that the solution path $\chi$ is piece-wise constant as a function of $\lambda$ [39, 41]. Therefore, in constructing a decreasing sequence of $\lambda$, it is only necessary to identify those "breakpoints" that lead to changes of the support set of $x_\lambda^*$, namely, either a new nonzero coefficient added or a previous nonzero coefficient removed.

The algorithm operates in an iterative fashion with an initial value $x^{(0)} = 0$. In each iteration, given a nonzero $\lambda$, we solve for $x$ satisfying $\partial F(x) = 0$. The first summand $f$ in Eq. 2.22 is differentiable: $\nabla f = A^T(Ax - y) = -c(x)$. The subgradient of $g(x) = ||x||_1$ is given by:

$$u(x) = \partial||x||_1 = \left\{ u \in \mathrm{R}^n : \begin{array}{c} u_1 = \mathrm{sgn}(x_i), x_i \neq 0 \\ u_i \in [-1, 1], x_i = 0 \end{array} \right\} \tag{2.24}$$

Thus, the solution to $\partial F(x) = 0$ is also the solution to the following equation:

$$c(x) = A^T b - A^T A x = \lambda u(x) \tag{2.25}$$

By the definition in Eq. 2.24, the sparse support set at each iteration is given by

$$I = \{i : |c_i^{(l)}| = \lambda\} \tag{2.26}$$

The algorithm then computes the update for $x^{(k)}$ in terms of the direction and the magnitude separately. Specifically, the update direction on the sparse support $d^{(k)}(I)$ is the solution to the following system:

$$A_I^T A_I d^{(k)}(I) = \mathrm{sgn}(c^{(k)}(I)) \tag{2.27}$$

where $A_I$ is a submatrix of $A$ that collects the column vectors of $A$ with respect to $I$, and $c^{(k)}(I)$ is a vector that contains the coefficients of $c^{(k)}$ with respect to $I$. For the coefficients whose indices are not in $I$, their update directions are manually set to zero. Along the direction indicated by $d^{(k)}$, there are two scenarios when an update on $x$ may lead to a breakpoint where the condition in Eq. 2.25 is violated. The first scenario occurs when an element of $c$ not in the support set would increase in magnitude beyond $\lambda$:

$$\gamma^+ = \min_{i \notin I} \left\{ \frac{\lambda - c_i}{1 - a_i^T A_I d^{(k)}(I)}, \frac{\lambda + c_i}{1 + a_i^T A_I d^{(k)}(I)} \right\} \tag{2.28}$$

The index that achieves $\gamma^+$ is denoted as $i^+$. The second scenario occurs when an element of $c$ in the support set $I$ crosses zero, violating the sign agreement:

$$\gamma^- = \min_{i \in I} \{-xi/di\} \tag{2.29}$$

The index that achieves $\gamma^-$ is denoted as $i^-$. Hence, the homotopy algorithm marches to the next breakpoint, and updates the sparse support set by either appending $I$ with $i^+$ or removing $i^-$:

$$x^{(k+1)} = x^{(k)} + \min\{\gamma^+, \gamma^-\} d^{(k)} \tag{2.30}$$

The algorithm terminates when the relative change in $x$ between consecutive iterations is sufficiently small. Algorithm 2 summarizes an implementation of the Homotopy methods [3].

**Algorithm 2.2:** Homotopy

**input** : A full rank matrix $A = [a_1, \cdots, a_n] \in \mathrm{R}^{m \times n}$ , $m < n$, a vector $b \in \mathrm{R}^m$ , initial Lagrangian parameter $\lambda = 2||A^T b||_\infty$

Initialization: $k \leftarrow 0$. Find the first support index:
$i = \arg\min_{j=1}^{n} ||a_j^T b||$, $I = \{i\}$ ;

**repeat**

- $k \leftarrow k + 1$ ;
- Solve for the update dictionary $d^{(k)}$ in Eq. 2.27 ;
- Compute the sparse support updates in Eq. 2.28 and Eq. 2.29:
- $\gamma^* \leftarrow \min\{\gamma^+, \gamma^-\}$ ;
- Update $x^{(k)}$, $I$, and $\lambda \leftarrow \lambda - \gamma^*$.

**until** *stopping criteria is satisfied.*;

**output**: $x^* \leftarrow x^{(k)}$

In overall, solving Eq. 2.27 using a Cholesky factorization and the addition/removal of the sparse support elements dominate the computation. Since one can keep track

[3] A MATLAB implementation can be found at `http://users.ece.gatech.edu/~sasif/homotopy/`.

of the rank-1 update of $A_I^T A_I$ in solving Eq. 2.27 using $O(m^2)$ operations in each iteration, the computational complexity of the homotopy algorithm is $O(km^2+kmn)$.

It has been shown that Homotopy shares some connections with two greedy $l_1$-min approximations: least angle regression (LARS) [41] and polytope faces pursuit (PFP) [42]. For instance, if the coefficient vector $x$ has at most $k$ non-zero components with $k \ll n$, all three algorithms can recover it in $k$ iterations. On the other hand, LARS never removes indices from the sparse support set during the iteration, while Homotopy and PFP have mechanisms to remove coefficients from the sparse support. More importantly, Homotopy provably solves $l_1$-min in Eq. 2.9, while LARS and PFP are only approximate solutions.

**Iterative Shrinkage-Thresholding (IST) Methods**

Although Homotopy employs a more efficient iterative update rule that only involves operations on those submatrices of $A$ corresponding to the support sets of $x$, it may not be as efficient when the sparsity $k$ and the observation dimension $m$ grow proportionally with the signal dimension $n$. In such scenarios, one can show that the worst-case computational complexity is still bounded by $O(n^3)$. In this section, we discuss Iterative Shrinkage-Thresholding (IST) methods [43, 44], whose implementation mainly involves simple operations such as vector algebra and matrix-vector multiplications. This is in contrast to most past methods that all involve expensive operations such as matrix factorization and solving linear least squares problems.

Concisely, IST considers Eq. 2.9 as a special case of the following composite ob-

jective function:

$$\min_x F(x) = f(x) + \lambda g(x), \tag{2.31}$$

where $f : \mathrm{R}^n \to \mathrm{R}$ is a smooth and convex function, and $g : \mathrm{R}^n \to \mathrm{R}$ as the regularization term is bounded from below but not necessarily smooth nor convex. For $l_1$-min in particular, $g$ is also separable, that is,

$$g(x) = \sum_{i=1}^{n} g_i(x_i) \tag{2.32}$$

Clearly, let $f(x) = \frac{1}{2}||b - Ax||_2^2$ , $g(x) = ||x||_1$. Then the objective function in Eq. 2.31 becomes the unconstrained basis pursuit de-noising (BPDN) problem [45].

The update rule to minimize Eq. 2.31 is computed using a second-order approximation of $f$ [46, 47]:

$$\begin{aligned} x^{(k+1)} &= \arg\min_x \{ f(x^{(k)}) + (x - x^{(k)})^T \nabla f(x^{(k)}) \\ &\qquad + \tfrac{1}{2}||x - x^{(k)}||_2^2 \cdot \nabla^2 f(x^{(k)}) + \lambda g(x) \} \\ &= \arg\min_x \{ (x - x^{(k)})^T \nabla f(x^{(k)}) \\ &\qquad + \tfrac{\alpha^{(k)}}{2}||x - x^{(k)}||_2^2 + \lambda g(x) \} \\ &= \arg\min_x \{ \tfrac{1}{2}||x - u^{(k)}||_2^2 + \tfrac{\lambda}{\alpha^{(k)}} g(x) \}, \\ &= G_{\alpha^{(k)}}(x^{(k)}) \end{aligned} \tag{2.33}$$

where

$$u^{(k)} = x^{(k)} - \frac{1}{\alpha^{(k)}} \nabla f(x^{(k)}) \tag{2.34}$$

In Eq. 2.33, the Hessian $\nabla^2 f(x^{(k)})$ is approximated by a diagonal matrix $\alpha^{(k)} I$.

If we replace $g(x)$ in Eq. 2.33 by the $l_1$-norm $||x||_1$, which is a separable function, then $G_{\alpha^{(k)}}(x^{(k)})$ has a closed-form solution with respect to each component:

$$\begin{aligned} x_i^{(k+1)} &= \arg\min_{x_i}\{\tfrac{(x_i-u_i^{(k)})^2}{2} + \tfrac{\lambda|x_i|}{\alpha^{(k)}}\} \\ &= \text{soft}\left(u_i^{(k)}, \tfrac{\lambda}{\alpha^{(k)}}\right), \end{aligned} \tag{2.35}$$

where

$$\begin{aligned} \text{soft}(u,a) &= \text{sgn}(u)\max\{|u|-a, 0\} \\ &= \begin{cases} \text{sgn}(u)(|u|-a) & \text{if } |u|>a \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.36}$$

is the soft-thresholding or shrinkage function [48].

There are two free parameters in Eq. 2.33, namely, the regularizing coefficient $\lambda$ and the coefficient $\alpha^{(k)}$ that approximates the Hessian matrix $\nabla^2 f$. Different strategies for choosing these parameters have been proposed. Since $\alpha I$ mimics the Hessian $\nabla^2 f$, we require that $\alpha^{(k)}(x^{(k)} - x^{(k-1)}) \approx \nabla f(x^{(k)}) - \nabla f(x^{(k-1)})$ in the least-squares sense. Hence,

$$\begin{aligned} \alpha^{(k+1)} &= \arg\min_{\alpha} ||\alpha(x^{(k)} - x^{(k-1)}) \\ &- \nabla f(x^{(k)}) - \nabla f(x^{(k-1)})||_2^2 \\ &= \tfrac{(x^{(k)}-x^{(k-1)})^T(\nabla f(x^{(k)})-\nabla f(x^{(k-1)})}{(x^{(k)}-x^{(k-1)})(x^{(k)}-x^{(k-1)})} \end{aligned} \tag{2.37}$$

For choosing $\lambda$, instead of using a fixed value, several papers have proposed a continuation strategy [44, 49], in which Eq. 2.33 is solved for a decreasing sequence of $\lambda$.

As mentioned in Section 2.1.2, Eq. 2.33 recovers the optimal $1_1$-min solution when $\lambda \to 0$.

The IST algorithm is summarized in Algorithm 3 [4].

**Algorithm 2.3:** Iterative Shrinkage-Thresholding (IST)

**input** : A full rank matrix $A \in \mathrm{R}^{m \times n}$ , $m < n$, a vector $b \in \mathrm{R}^m$ , Lagrangian parameter $\lambda_0$, initial values for $x^{(0)}$ and $\alpha^{(0)}$, $k \leftarrow 0$

Generate a reducing sequence $\lambda_0 > \lambda_1 > \cdots > \lambda_N$ ;

**for** $i = 0, 1, \cdots, N$ **do**

  $\lambda \leftarrow \lambda_i$ ;

  **repeat**

    $k \leftarrow k + 1$ ;

    $x^{(k)} \leftarrow G(x^{(k-1)})$ ;

    Update $\alpha^{(k)}$ using Eq. 2.37.

  **until** *The objective function* $F(x^{(k)})$ *decreases.*;

**end**

**output**: $x^* \leftarrow x^{(k)}$

### Proximal Gradient (PG) Methods

PG algorithms represent another class of algorithms that solve convex optimization problems of the form in Eq. 2.31. Assume that our cost function $F(\cdot)$ can be decomposed as the sum of two functions $f$ and $g$, where $f$ is a smooth convex function with Lipschitz continuous gradient, and $g$ is a continuous convex function. The principle behind these algorithms is to iteratively form quadratic approximations $Q(x, y)$ to $F(x)$ around a carefully chosen point $y$, and to minimize $Q(x, y)$ rather than the original cost function $F$.

[4] A MATLAB implementation called Sparse Reconstruction by Separable Approximation (SpaRSA) [47] is available at `http://www.lx.it.pt/~mtf/SpaRSA/`.

For our problem, we define $g(x) = ||x||_1$ and $f(x) = \frac{1}{2}||b - Ax||_2^2$. We note that $\nabla f(x) = A^T(Ax - b)$ is Lipschitz continuous with Lipschitz constant $L_f = ||A||^2$. Define $Q(x, y)$ as:

$$Q(x, y) = f(y) + \langle f(y), x - y\rangle + \frac{L_f}{2}||x - y||^2 + \lambda g(x) \tag{2.38}$$

It can be shown that $F(x) \leq Q(x, y)$ for all $y$, and

$$\arg\min_x Q(x, y) = \arg\min_x \left\{ \lambda g(x) + \frac{L_f}{2}||x - u||_2^2 \right\} \tag{2.39}$$

where $u = y - \frac{1}{L_f}\nabla f(y)$ by the same trick used in Eq. 2.33. For the $1_1$-min problem in Eq. 2.9, Eq. 2.39 has a closed-form solution given by the soft-thresholding function:

$$\arg\min_x Q(x, y) = \text{soft}\left(u, \frac{\lambda}{L_f}\right) \tag{2.40}$$

However, unlike the iterative thresholding algorithm described earlier, we use a smoothed computation of the sequence $y_k$. It has been shown that choosing

$$y^{(k)} = x^{(k)} + \frac{t_{k-1} - 1}{t_k}\left(x^{(k)} - x^{(k-1)}\right) \tag{2.41}$$

where $\{t_k\}$ is a positive real sequence satisfying $t_k^2 - t_k \leq t_{k-1}^2$, achieves an accelerated non-asymptotic convergence rate of $O(k^{-2})$ [50, 46]. To further accelerate the convergence of the algorithm, one can also make use of the continuation technique described in Section 2.1.2. Finally, for large problems, it is often computationally expensive to directly compute $Lf = ||A||^2$. A backtracking line-search strategy [46] can be used to generate a scalar sequence $\{L_k\}$ that approximates $L_f$. We define

$$Q_L(x, y) = f(y) + (x - y)^T\nabla f(y) + \frac{L}{2}||x - y||^2 + \lambda g(x) \tag{2.42}$$

Suppose that $\eta > 1$ is a pre-defined constant. Then, given $y^{(k)}$ at the kth iteration, we set $L_k = \eta^j L_{k-1}$, where $j$ is the smallest nonnegative integer such that the following inequality holds:

$$F(G_{L_k}(y^{(k)})) \leq Q_{L_k}(G_{L_k}(y^{(k)}), y^{(k)}), \tag{2.43}$$

where $G_L(y) = \arg\min_x Q_L(x, y) = \text{soft}\left(u, \frac{\lambda}{L}\right)$ for $u = y - \frac{1}{L}\nabla f(y)$.

The algorithm, named FISTA in [46], is summarized as Algorithm 4 [5].

**Algorithm 2.4:** Fast Iterative Shrinkage-Threshold Algorithm (FISTA)

**input** : $b \in \mathrm{R}^m$, $A \in \mathrm{R}^{m \times n}$
Set $x^{(0)} \leftarrow 0$, $x^{(1)} \leftarrow 0$, $t_0 \leftarrow 1$, $t_1 \leftarrow 1$, $k \leftarrow 1$ ;
Initialize $L_0$, $\lambda_1$, $\beta \in (0, 1)$, $\bar{\lambda} > 0$ ;
**while** *not converged* **do**
  $y^{(k)} \leftarrow x^{(k)} + \frac{t_{k-1}-1}{t_k}(x^{(k)} - x^{(k-1)})$ ;
  Update $L_k$ using Eq. 2.43 with $y^{(k)}$;
  $u^{(k)} \leftarrow y^{(k)} - \frac{1}{L_k}A^T(Ay^{(k)} - b)$;
  $x^{(k+1)} \leftarrow \text{soft}\left(u^{(k)}, \frac{\lambda}{L_k}\right)$;
  $t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$ ;
  $\lambda_{k+1} \leftarrow \max(\beta\lambda_k, \bar{\lambda})$ ;
  $k \leftarrow k + 1$
**end**
**output**: $x^* \leftarrow x^{(k)}$

### Augmented Lagrange Multiplier (ALM) Methods

Lagrange multiplier methods are a popular class of algorithms in convex programming. The basic idea is to eliminate equality constraints and instead add a penalty term to the cost function that assigns a very high cost to infeasible points. Augmented

[5] An implementation of FISTA is provided at `http://www.eecs.berkeley.edu/~yang/software/l1benchmark/index.html`

Lagrange Multiplier (ALM) methods differ from other penalty-based approaches by simultaneously estimating the optimal solution and Lagrange multipliers in an iterative fashion.

We consider the general $1_1$-min problem in Eq. 2.9 with the optimal solution $x^*$. The corresponding augmented Lagrange function is given by

$$L_\mu(x, y) = ||x||_1 + \langle y, b - Ax \rangle + \frac{\mu}{2}||b - Ax||_2^2 \tag{2.44}$$

where $\mu > 0$ is a constant that determines the penalty for infeasibility, and $y$ is a vector of Lagrange multipliers. Let $y^*$ be a Lagrange multiplier vector satisfying the second-order sufficiency conditions for optimality (see [51] for more details). Then, for sufficiently large $\mu$, it can be shown that

$$x^* = \arg\min_x L_\mu(x, y^*) \tag{2.45}$$

The main problem with this solution is that $y^*$ is unknown in general. Furthermore, the choice of $\mu$ is not straightforward from the above formulation. It is clear that to compute $x^*$ by minimizing $L_\mu(x, y)$, we must choose $y$ close to $y^*$ and set $\mu$ to be a very large positive constant. The following iterative procedure has been proposed in [51] to simultaneously compute $y^*$ and $x^*$ :

$$\begin{cases} x^{(k+1)} = \arg\min_x L_{\mu_k}(x, y^(k)) \\ y^{(k+1)} = y^{(k)} + \mu_k(b - Ax^{(k+1)}) \end{cases} \tag{2.46}$$

where $\{\mu_k\}$ is a monotonically increasing positive sequence. We note that the first step in the above procedure is itself an unconstrained convex optimization problem. Thus, the above iterative procedure is computationally efficient only if it is easier to

minimize the augmented Lagrangian function compared to solving the the original constrained optimization problem in Eq. 2.9 directly.

We focus our attention on solving the first step of Eq. 2.46 for the $l_1$-min problem. Although it is not possible to obtain a closed-form solution, the cost function has the same form as described in Eq. 2.31. Furthermore, the quadratic penalty term is smooth and has a Lipschitz continuous gradient. Therefore, we can solve it efficiently using proximal gradient methods (e.g., FISTA) described in Section 2.1.2.

The entire algorithm is summarized as Algorithm 5, where $\tau$ represents the largest eigenvalue of the matrix $A^TA$, and $\rho > 1$ is a constant.

**Algorithm 2.5:** Augmented Lagrange Multiplier (ALM)

**input** : $b \in \mathrm{R}^m$, $A \in \mathrm{R}^{m \times n}$
**while** *not converged* $(k = 1, 2, \cdots)$ **do**
  $t_1 \leftarrow 1$, $z_1 \leftarrow x^{(k)}$, $u^{(1)} \leftarrow x^{(k)}$ ;
  **while** *not converged* $(l = 1, 2, \cdots)$ **do**
    $u^{(l+1)} \leftarrow \text{soft}\left(z_l - \frac{1}{\tau}A^T\left(Az_l - b - \frac{1}{\mu_k}y^{(k)}\right), \frac{1}{\mu_k\tau}\right)$;
    $t_{l+1} \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4t_l^2}\right)$ ;
    $z_{l+1} \leftarrow u^{(l+1)} + \frac{t_l - 1}{t_{l+1}}(u^{(l+1)} - u^{(l)})$ ;
  **end**
  $x^{(k+1)} \leftarrow u^{(l+1)}$;
  $y^{(k+1)} \leftarrow y^{(k)} + \mu_k(b - Ax^{(k+1)})$;
  $\mu_{k+1} \leftarrow \rho \cdot \mu_k$
**end**
**output**: $x^* \leftarrow x^{(k)}$

Algorithm 5 summarizes the ALM method applied to the primal problem in Eq. 2.9, which is referred to as Primal ALM (PALM). Interestingly, the principles of ALM

can also be applied to its dual problem [52]:

$$\max_{y} b^T y \;\; subject\ to\ A^T y \in B_1^\infty, \tag{2.47}$$

where $B_1^\infty = \{x \in \mathrm{R}^n : ||x||_\infty \leq 1\}$. Under certain circumstances, this may result in a more computationally efficient algorithm.

Simple computation shows that DALM solves the following problem:

$$\begin{aligned} \min_{y,x,z} \quad & -b^T y - x^T(z - A^T y) + \tfrac{\beta}{2}||z - A^T y||_2^2 \\ subject\ to \quad & z \in B_1^\infty \end{aligned} \tag{2.48}$$

Here, $x$ as the primal variable becomes the associated Lagrange multiplier in the dual space [52]. Since it is difficult to solve the above problem simultaneously with respect to $y, x$ and $z$, we adopt a strategy by alternately updating the primal variable $x$ and the dual variables $y$ and $z$.

On one hand, for $x = x^{(k)}$ and $y = y^{(k)}$ , the minimizer $z_{k+1}$ with respect to $z$ is given by

$$z_{k+1} = P_{B_1^\infty}(A^T y^{(k)} + x^{(k)}/\beta), \tag{2.49}$$

where $P_{B_1^\infty}$ represents the projection operator onto $B_1^\infty$ . On the other hand, given $x = x^{(k)}$ and $z = z_{k+1}$, the minimization with respect to $y$ is a least squares problem, whose solution is given by the solution to the following equation:

$$\beta A A^T y = \beta A z_{k+1} - (A x^{(k)} - b). \tag{2.50}$$

Suppose that $AA^T$ is invertible. Then, we directly use its inverse to solve Eq. 2.50. However, for large scale problems, this matrix inversion can be computationally expensive. Therefore, in such cases, we can approximate the solution with one step of

the Conjugate Gradient algorithm in the $y$ direction at each iteration, as proposed in [52].

The basic iteration of the DALM algorithm is given by [6]

$$\begin{cases} z_{k+1} = P_{A_1^\infty}(A^T y^{(k)} + x^{(k)}/\beta), \\ y^{(k+1)} = (AA^T)^{-1}(Az_{k+1} - (Ax^{(k)} - b)/\beta), \\ y^{(k+1)} = x^{(k)} - \beta(z_{k+1} - A^T y^{(k+1)}), \end{cases} \tag{2.51}$$

### 2.1.3 Dictionary Learning

An overcomplete dictionary that leads to sparse representations can either be chosen as a prespecified set of functions (e.g. Wavelet tramsforms) or designed by adapting its content to fit a given set of signal examples. Choosing a prespecified transform matrix is appealing because it is simpler. Also, in many cases it leads to simple and fast algorithms for the evaluation of the sparse representation. The success of such dictionaries in applications depends on how suitable they are to sparsely describe the signals in question.

Dictionary training is a recent approach that has been strongly influenced by the latest advances in sparse representation theory and algorithms [53]. The aim of learning dictionary elements is to out-perform commonly used predetermined dictionaries by using methods that adapt dictionaries for special classes of signals. The most recent training methods focus on $l_0$ and $l_1$ sparsity measures, which lead to simple formulations and enable the use of recently developed efficient sparse-coding techniques [45, 43, 38, 54].

[6] The PALM and DALM algorithms in MATLAB can be downloaded from `http://www.eecs.berkeley.edu/~yang/software/l1benchmark/index.html`.

Starting with the work of Olshausen and Field [55], dictionary training approaches have become a popular approach to dictionary design. The method in [55] is based on maximizing the likelihood (ML) that a natural image, $b$, arises from the overcomplete dictionary, $D$, when the generative image model is considered as sparse image decomposition into dictionary elements. Therefore, the ML method solves the optimization problem $D^* = \max_D P(b|D)$, for $b = D \cdot x$. The optimization is solved in two iterative steps: the sparse coding step, where the dictionary is kept fixed and the sparse coefficient vector, $x$, that best approximates the image is found; and a dictionary update step, where $x$ is kept fixed and the dictionary is updated to maximize the objective maximum likelihood function using gradient descent.

### Method of Optimal Directions

The Method of Optimal Directions (MOD) was introduced by Engan et al. in 1999 [56], and was one of the first methods to implement what is known today as a sparsification process. Given a set of examples $B = [b_1 b_2 \cdots b_n]$, the goal of the MOD is to find a dictionary $D$ and a sparse matrix $X$ which minimize the representation error,

$$\arg\min_{D,X} ||B - DX||_F^2 \quad subject\ to\ ||x_i||_0 \leq T, \forall i, \tag{2.52}$$

where $\{x_i\}$ represent the columns of $X$, and the $l_0$ sparsity measure $||\cdot||_0$ counts the number of non-zeros in the representation. The resulting optimization problem is combinatorial and highly non-convex, and thus we can only hope for a local minimum at best. Similar to other training methods, the MOD alternates sparse-coding and dictionary update steps. The sparse-coding is performed for each signal individually using any standard technique. For the dictionary update, 2.52 is solved via the analytic solution of the quadratic problem, given by $D = BX^+$ with $X^+$ denoting the

Moore-Penrose pseudo-inverse.

The MOD typically requires only a few iterations to converge, and is overall a very effective method. The method suffers, though, from the relatively high complexity of the matrix inversion. Several subsequent works have thus focused on reducing this complexity, leading to more efficient methods.

### Union of Orthobases

Training a union-of-orthobases dictionary was proposed in 2005 by Lesage et al. [57] as a means of designing a dictionary with reduced complexity and which could be more efficiently trained. The process also represents one of the first attempts at training a structured overcomplete dictionary –a tight frame in this case. The model suggests training a dictionary which is the concatenation of $k$ orthogonal bases, so $D = [D_1 D_2 \cdots D_k]$ with the $\{D_i\}$ unitary matrices. Sparse-coding over this dictionary can be performed efficiently through a Block Coordinate Relaxation (BCR) technique [58].

A drawback of this approach is that the proposed model itself is relatively restrictive, and in practice it does not perform as well as more flexible structures. Interestingly, there is a close connection between this structure and the more powerful Generalized PCA (GPCA) model, described next. The GPCA also arises from a union of orthogonal spaces model, though it deviates from the classical sparse representation paradigm. Identifying such relations could thus prove valuable in enabling a merge between the two forces.

### Generalized PCA

Generalized PCA, introduced in 2005 by Vidal, Ma and Sastry [59], offers a different and very interesting approach to overcomplete dictionary design. The GPCA view is basically an extension of the original PCA formulation, which approximates a set of examples by a low-dimensional subspace. In the GPCA setting, the set of examples is modeled as the union of several low-dimensional subspaces –perhaps of unknown number and variable dimensionality –and the algebraic-geometric GPCA algorithm determines these subspaces and fits orthogonal bases to them.

The GPCA viewpoint differs from the sparsity model described as $b = Dx$, as each example in the GPCA setting is represented using only one of the subspaces; thus, atoms from different subspaces cannot jointly represent a signal. This property has the advantage of limiting over-expressiveness of the dictionary, which characterizes other overcomplete dictionaries; on the other hand, the dictionary structure may be too restrictive for more complex natural signals.

A unique property of the GPCA is that as opposed to other training methods, it can detect the number of atoms in the dictionary in certain settings. Unfortunately, the algorithm may become very costly this way, especially when the amount and dimension of the subspaces increases. Indeed, intriguing models arise by merging the GPCA viewpoint with the classical sparse representation viewpoint: for instance, one could easily envision a model generalizing 2.52 where several distinct dictionaries are allowed to co-exists, and every signal is assumed to be sparse over exactly one of these dictionaries.

### The K-SVD Algorithm

The desire to efficiently train a generic dictionary for sparse signal representation led Aharon, Elad and Bruckstein to develop the K-SVD algorithm in 2006 [60]. The algorithm aims at the same sparsification problem as the MOD in Eq. 2.52, and employs a similar block-relaxation approach. The main contribution of the K-SVD is that the dictionary update, rather than using a matrix inversion, is performed atom-by-atom in a simple and efficient process. Further acceleration is provided by updating both the current atom and its associated sparse coefficients simultaneously. The result is a fast and efficient algorithm which is less demanding than the MOD.

The K-SVD algorithm takes its name from the Singular-Value-Decomposition (SVD) process that forms the core of the atom update step, and which is repeated K times, as the number of atoms. For a given atom k, the quadratic term in Eq. 2.52 is rewritten as

$$\left\| B - \sum_{j \neq k} d_j x_j^T - d_k x_k^T \right\|_F^2 = ||E_k - d_k x_k^T||_F^2 \tag{2.53}$$

where $\{x_j^T\}$ are the rows of $X$, and $E_k$ is the residual matrix. The atom update is obtained by minimizing 2.53 for $d_k$ and $x_k^T$ via a simple rank-1 approximation of $E_k$ . To avoid introduction of new non-zeros in $X$, the update process is performed using only the examples whose current representations use the atom $d_k$ .

In practice, the K-SVD is an effective method for representing small signal patches. However, the K-SVD, as well as the MOD, suffer from a few common weaknesses. The high non-convexity of the problem means that the two methods will get caught in

local minima or even saddle points. Also, the result of the training is a non-structured dictionary which is relatively costly to apply, and therefore these methods are suitable for signals of relatively small size. In turn, in recent years several parametric dictionary training methods have begun to appear, and aim to address these issues by importing the strengths of analytic dictionaries to the world of example-based methods.

### 2.1.4 Applications in Computer Vision

In this section, we are going to introduce compressed sensing and sparse representation based methods that are applied to various problems in computer vision.

#### Single Pixel Camera

As we have described above, CS combines sampling and compression into a single non-adaptive linear measurement process [61]. Rather than measuring pixel samples of the scene under view, the authors of [1] measure inner products between the scene and a set of test functions. Interestingly, random test functions play a key role, making each measurement a random sum of pixel values taken across the entire image. When the scene under view is compressible by an algorithm like JPEG or JPEG2000, the CS theory enables us to stably reconstruct an image of the scene from fewer measurements than the number of reconstructed pixels. In this manner they achieve sub-Nyquist image acquisition.

Following this observation, they propose 'single-pixel' CS camera architecture that is basically an optical computer (comprising a DMD, two lenses, a single photon detector, and an analog-to-digital (A/D) converter) that computes random linear

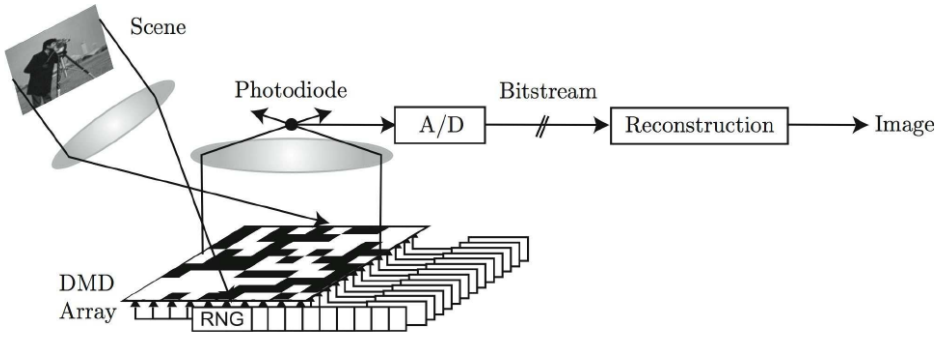



**Figure 2.1**: Compressed Sensing (CS) camera block diagram [1].

measurements of the scene under view (Fig. 2.1). The image is then recovered or processed from the measurements by a digital computer. The camera design reduces the required size, complexity, and cost of the photon detector array down to a single unit, which enables the use of exotic detectors that would be impossible in a conventional digital camera. The random CS measurements also enable a tradeoff between space and time during image acquisition. Finally, since the camera compresses as it images, it has the capability to efficiently and scalably handle high-dimensional data sets from applications like video and hyperspectral imaging.

Mathematically, the single-pixel camera sequentially measures the inner products $b[m] = \langle f, \phi_m \rangle$ between an $N$-pixel sampled version $f$ of the incident light-field from the scene under view and a set of two-dimensional (2-D) test functions $\{\phi_m\}$ [1]. As shown in Figure 2.1, the light-field is focused by a biconvex lens onto a digital micromirror device consisting of an array of $N$ tiny mirrors.

Each mirror corresponds to a particular pixel in $f$ and $\phi_m$ and can be independently


**Figure 2.2**: (a) $256 \times 256$ conventional image, (b) Single-pixel camera reconstructed image from $M = 1300$ random measurements [2].

oriented either towards the lens (corresponding to a one at that pixel in $\phi_m$ ) or away from the lens (corresponding to a zero at that pixel in $\phi_m$ ). The reflected light is then collected by the lens and focused onto a single photon detector (the single pixel) that integrates the product $f[n]\phi_m[n]$ to compute the measurement $b[m] = \langle f, \phi_m \rangle$ as its output voltage. This voltage is then digitized by an A/D converter. Values of $\phi_m$ between zero and one can be obtained by dithering the mirrors back and forth.

To compute CS randomized measurements $b = \Phi f$ as in 2.3, they set the mirror orientations $\phi_m$ randomly using a pseudorandom number generator, measure $b[m]$, and then repeat the process $M$ times to obtain the measurement vector $b$. After they collect the measurements, they obtain the sparse representation $x$ by solving the problem in Eq. 2.6 and reconstruct the the image $f^* = \Psi x^*$. Figure 2.2 illustrates a target object (a black-and-white printout of an "R") $f$ and reconstructed image $f^*$ taken by the single-pixel camera prototype in Figure 2.1 using $N = 256 \times 256$ and $M = N/50$.

**Classification via Sparse Representation**

Automatic face recognition remains to be one of the most visible and challenging application domains in vision. In this section, we will see how sparse representation and sparse error correction can be used to achieve robust face recognition in scenarios where well-controlled training images can be collected [11, 3]. The key idea is a wise choice of dictionary: representing the test signal as a sparse linear combination of the training signals themselves. We will see how this approach leads to simple and effective algorithms for face recognition. In turn, the face recognition example reveals new theoretical phenomena in sparse representation that may at first seem surprising.

The authors of [11] assumes that there is an access to well-aligned training images of each subject, taken under varying illumination. For a detailed explanation of how such images can be obtained, please see [62]. They stack the given $N_i$ training images from the $i$th class as columns of a matrix: $A_i = [a_{i,1}, a_{i,2}, \cdots, a_{i,N_i}] \in \mathrm{R}^{m \times N_i}$, each normalized to have unit $l_2$-norm. One classical observation from computer vision is that images of the same face under varying illumination lie near a special low-dimensional subspace [63], often called a face subspace. So, given a sufficiently expressive training set $A_i$ , a new image of subject $i$ taken under different illumination and also stacked as a vector $b \in \mathrm{R}^m$ can be represented as a linear combination of the given training: $b \approx A_i x_i$ for some coefficient vector $x_i \in \mathrm{R}^{N_i}$.

The problem becomes more challenging if the identity of the test sample is initially unknown. They define a new matrix $A$ for the entire training set as the concatenation

of the $N = \sum_i N_i$ training samples of all $c$ object classes

$$A = [A_1, A_2, \cdots, A_c] = [a_{1,1}, a_{1,2}, \cdots, a_{k,N_k}] \tag{2.54}$$

Then, the linear representation of $b$ can be rewritten in terms of all training samples as

$$b = Ax_0 \in \mathrm{R}^m \tag{2.55}$$

where $x_0 = [0, \cdots, 0, x_i^T, 0, \cdots, 0] \in \mathrm{R}^N$ is a coefficient vector whose entries are all zero except for those associated with the $i$th class. The special support pattern of this coefficient vector is highly informative for recognition: ideally, it precisely identifies the subject pictured. However, in practical face recognition scenarios, the search for such an informative coefficient vector $x_0$ is often complicated by the presence of partial corruption or occlusion: gross errors affect some fraction of the image pixels. In this case, they modify the above linear model in Eq. 2.55 as

$$b = b_0 + e_0 = Ax_0 + e_0 \tag{2.56}$$

where $e_0 \in \mathrm{R}^m$ is a vector of errors –a fraction $\rho$ of its entries are nonzero.

Thus, face recognition in the presence of varying illumination and occlusion can be treated as the search for a certain sparse coefficient vector $x_0$ , in the presence of a certain sparse error $e_0$. The number of unknowns in Eq. 2.56 exceeds the number of observations, and we cannot directly solve for $x_0$. However, under mild conditions [64], the desired solution $(x_0, e_0)$ is not only sparse, but also it is the sparsest solution to the system of Eq. 2.56

$$(x_0, e_0) = \arg\min ||x||_0 + ||e||_0 \;\; \textit{subject to} \;\; b = Ax + e \tag{2.57}$$

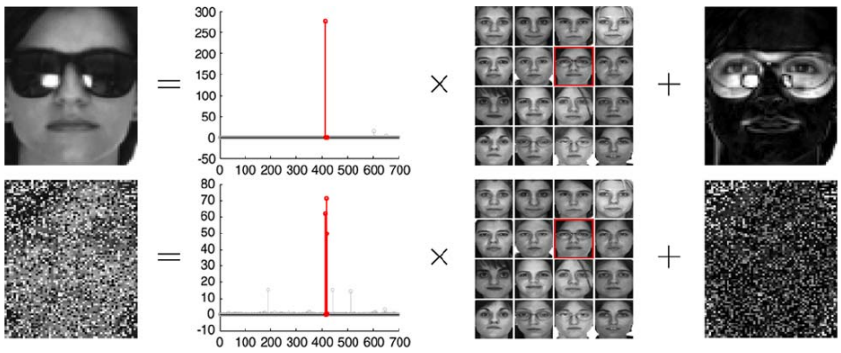



**Figure 2.3**: Overview of the face recognition approach [3].

Here, the $l_0$-"norm" $||\cdot||_0$ counts the number of nonzeros in a vector. Originally inspired by theoretical results on equivalence between $l_1$ -and $l_0$- minimizations [29, 65, 66], the authors of [11] proposed to seek this informative vector $x_0$ by solving the convex relaxation

$$\min ||x||_1 + ||e||_1 \quad subject\ to \quad b = Ax + e \tag{2.58}$$

where $||x||_1 = \sum_i |x_i|$. This work demonstrates empirically an interesting tendency of the $l_1$-minimizer: as visualized in Fig. 2.3, sparse representation separates the identity of the face (red coefficients) from the error due to corruption or occlusion.

Once the $l_1$-minimization problem has been solved (e.g., [45, 38]), classification (identifying the subject pictured) or validation (determining if the subject is present in the training database) can proceed by considering how strongly the recovered coefficients concentrate on any one subject (please refer to [11] for details). Here, we present only a few representative results; a more thorough empirical evaluation can be found in [11]. Fig. 2.4 (left) compares the recognition rate of this approach (labeled SRC) with several popular methods on the Extended Yale B Database [63] under varying

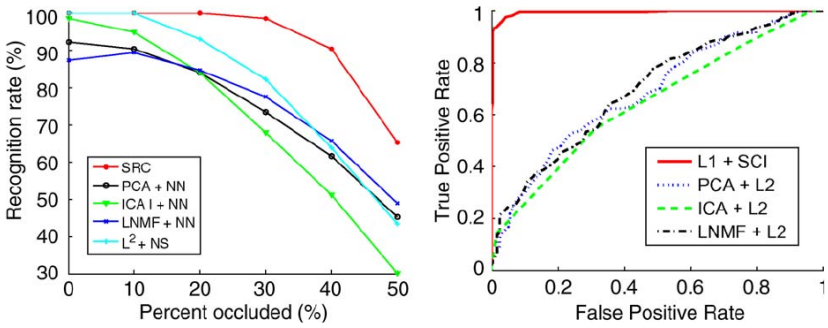



**Figure 2.4**: Face recognition and validataion [3].

levels of synthetic block occlusion.

Fig. 2.4 compares the sparsity-based approach outlined here with several popular methods from the literature: the principal component analysis (PCA) approach of [67], independent component analysis (ICA) architecture I [68], and local nonnegative matrix factorization (LNMF) [69]. The first method provides a standard baseline of comparison, while the latter two methods are more directly suited for occlusion, as they produce lower dimensional feature sets that are spatially localized. Fig. 2.4 (left) also compares to the nearest subspace method [70], which makes similar use of linear illumination models, but does not correct sparse errors.

The $l_1$-based approach achieves the highest overall recognition rate of the methods tested, with almost perfect recognition up to 30% occlusion and a recognition rate above 90% with 40% occlusion. Fig. 2.4 (right) shows the validation performance of the various methods, under 30% contiguous occlusion, plotted as a receiver operating characteristic (ROC) curve. At this level of occlusion, the sparsity-based method is

the only one that performs significantly better than chance. The performance under random pixel corruption is also strong (Fig. 2.3 (bottom)), with recognition rates above 90% even at 70% corruption.

The empirical results seem to demand a correspondingly strong theoretical justification. However, a more thoughtful consideration reveals that the underdetermined system of linear equation in Eq. 2.56 does not satisfy popular sufficient conditions for guaranteeing correct sparse recovery by $l_1$-minimization.

In face recognition, the columns of $A$ are highly correlated: they are all images of some face. As $m$ becomes large (i.e., the resolution of the image becomes high), the convex hull spanned by all face images of all subjects is only an extremely tiny portion of the unit sphere $S^{m-1}$ . For example, the images (in $96 \times 84$ resolution) in Fig. 2.3 lie on $S^{8063}$. The smallest inner product with their normalized mean is 0.723; they are contained within a spherical cap of volume $\leq 1.47 \times 10^{-229}$. These vectors are tightly bundled together as a "bouquet", whereas the standard pixel basis $\pm I$ with respect to which we represent the errors $e$ forms a "cross" in $\mathrm{R}^m$ , as illustrated in Fig. 2.5. The incoherence and restricted isometry properties that are so useful in providing performance guarantees for $l_1$-minimization therefore do not hold for the "cross-and-bouquet" matrix $[A \;\; I]$ [3]. Also, the density of the desired solution is not uniform either: $A$ is usually a very sparse nonnegative vector, but $e$ could be dense (with a fraction nonzeros close to one) and have arbitrary signs. Existing results for recovering sparse signals suggest that $l_1$-minimization may have difficulty in dealing with such signals, contrary to its empirical success in face recognition.

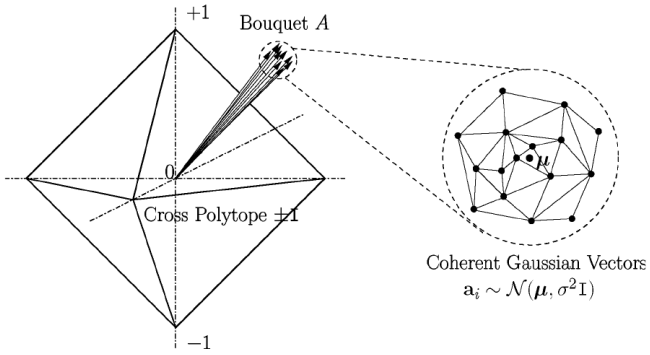



**Figure 2.5**: The "cross-and-bouquet" model [3].

In an attempt to better understand the face recognition example outlined above, the authors of [3] consider the more abstract problem of recovering such a nonnegative sparse signal $x_0 \in \mathrm{R}^N$ from highly corrupted observations $b \in \mathrm{R}^m$

$$b = Ax_0 + e0 \tag{2.59}$$

where $e_0 \in \mathrm{R}^m$ is a vector of errors of arbitrary magnitude. The model for $A \in \mathrm{R}^{m \times N}$ should capture the idea that it consists of small deviations about a mean, hence a "bouquet". This can be modeled by assuming the columns of $A$ are independent identically distributed (i.i.d.) samples from a Gaussian distribution

$$A = [a_1, a_2, \cdots, a_N] \in \mathrm{R}^{m \times N}, \quad a_i \sim_{i.i.d.} N\left(\mu, \frac{\nu^2}{m} I_m\right)$$
$$||\mu||_2 = 1, \quad ||\mu||_\infty \leq C_\mu m^{-1/2} \tag{2.60}$$

Together, the two assumptions on the mean force $\mu$ to remain incoherent with the standard basis (or "cross") as $m \to \infty$.

In [3], it has been proven that, as long as the bouquet is sufficiently tight, asymptotically $l_1$-minimization recovers any nonnegative sparse signal from almost any error

with support size less than 100%. This provides some theoretical confirmation to the strong empirical results observed in the face recognition example, especially in the presence of random corruption.

The face recognition approach described here assumes that the training images have been carefully controlled and that the number of samples per class is sufficiently large. Outside these operating conditions, and in particular when only a single sample per class is available, it should not be expected to perform well.

Although the cross-and-bouquet model explains much of the error correction ability of $l_1$ minimization, the discriminative power of the sparse representation still lacks rigorous mathematical justification. This remains a wide open topic for future investigation.

**Image Denoising**

In the previous section, we have examined an application in vision and machine learning in which a sparse representation in an overcomplete dictionary consisting of the samples themselves yielded semantic information. This is an extremely useful idea for clustering and classification, especially for problems such as face recognition where the data have linear or piecewise linear structure. However, for applications such as inpainting or denoising, the identity of the given training samples is less important. In such applications, it may be possible to learn more relevant dictionaries by optimizing a task-specific objective function.

As detailed in the previous sections, sparse modeling calls for constructing efficient

representations of data as a (often linear) combination of a few typical patterns (atoms) learned from the data itself. As we have explained in previous sections, significant contributions to the theory and practice of learning such collections of atoms (usually called dictionaries or codebooks), e.g., [60, 56, 55], and of representing the actual data in terms of them, e.g., [45, 43, 41], have been developed in recent years, leading to state-of-the-art results in many signal and image processing tasks [71, 72, 73, 74].

The dictionary plays a critical role, and it has been shown that learned dictionaries significantly outperform off-the-shelf ones such as wavelets [53]. Current techniques for obtaining such dictionaries mostly involve their optimization in terms of the task to be performed, e.g., representation [56], denoising [60, 74], and classification [73].

Let $B \in \mathrm{R}^{m \times N}$ be a set of $N$ column data vectors $b_j \in \mathrm{R}^m$ (e.g., image patches), and $D \in \mathrm{R}^{m \times K}$ be a dictionary of $K$ atoms represented as columns $d_k \in \mathrm{R}^m$. Each data vector $b_j$ will have a corresponding vector of reconstruction coefficients $x_j \in \mathrm{R}^K$, which we will treat as columns of a matrix

$$X = [x_1 \ \cdots \ x_N] \in \mathrm{R}^{K \times N} : \tag{2.61}$$

The goal of sparse modeling is to design a dictionary $D$ such that $B \simeq DX$ with $||x_j||_0$ sufficiently small (usually below some threshold) for all or most data samples $b_j$.

As we have explained in Section 2.1.3, let's begin with the standard $l_0$ or $l_1$ penalty modeling problem

$$(X^*, D^*) = \arg\min_{X,D} ||B - DX||_F^2 + \lambda ||X||_p \tag{2.62}$$

where $||\cdot||_F$ denotes Frobenius norm and $p = 0, 1$. The cost function to be minimized in Eq. 2.62 consists of a quadratic fitting term and an $l_0$ or $l_1$ regularization term for each column of $X$, the balance of the two being defined by the penalty parameter $\lambda$ . The $l_1$-norm can be used as an approximation to $l_0$, making the problem convex in $X$ while still encouraging sparse solutions [37]. While for reconstruction the $l_0$ penalty is found to often produce better results, $l_1$ leads to more stable active sets and is preferred for the classification tasks.

Since Eq. 2.62 is not simultaneously convex in $\{X, D\}$, coordinate-descent-type optimization techniques have been proposed [60, 56]. These approaches have been extended for multiscale dictionaries and color images in [74], leading to state-of-the-art results. In Fig. 2.6 an example of color image denosing with this approach is given. For more examples, comparisons, and applications in image demosaicing, image inpainting, and image denoising, please refer [22, 74]. An example of a learned dictionary of $K = 256$ atoms is also shown in Fig. 2.6. It is important to note that for image denoising, overcomplete dictionaries are used $K > m$, and the patch sizes vary from $7 \times 7$, $m = 49$, to $20 \times 20$, $m = 400$ (for color image denoising), with a sparsity of about one tenth of the signal dimension $m$.

## 2.2 Visual Sensor Networks

### 2.2.1 Overview

For many years now, networks of cameras have been used for surveillance and security. These networked cameras, which are unable to process any data locally, send their individual streams to a centralized location and rely upon human vision and

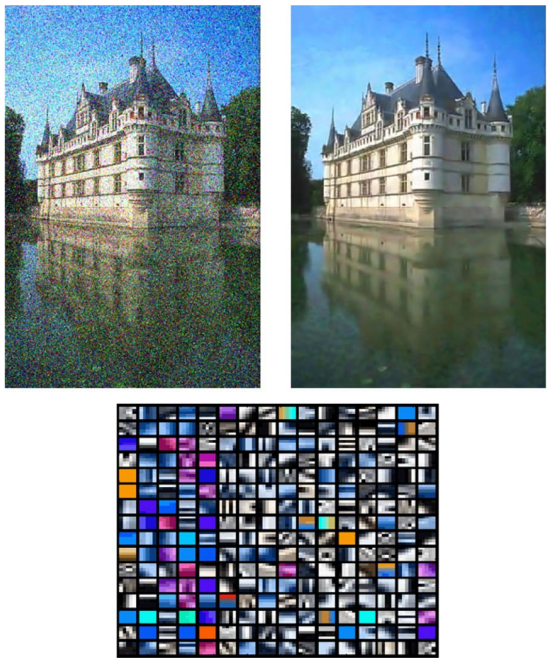


**Figure 2.6**: Image denoising via sparse modeling and dictionary learned from a standard set of color images.

**Figure 2.7**: The nodes in VSNs consist of image sensor, embedded processor and wireless transceiver.

cognition to detect events and anomalies. Advances in computational technologies have allowed for much improvement in the hardware of these systems, such as higher picture quality and motorization capabilities. However, only recently there have been attempts to integrate the latest research developments in human and computer vision into current sensor technologies.

With the advances in image sensor technology, low-power image sensors have appeared in a number of products, such as cell phones, toys, computers, and robots. Furthermore, recent developments in sensor networking and distributed processing have encouraged the use of image sensors in these networks, which has resulted in a new ubiquitous paradigm - visual sensor networks [75]. Visual sensor networks (VSNs) consist of small visual sensor nodes called camera nodes, which integrate the image sensor, embedded processor, and wireless transceiver (Figure 2.7). Following the trends in low-power processing, wireless networking, and distributed sensing, visual sensor networks have developed as a new technology with a number of potential applications, ranging from security to monitoring to telepresence.

In a visual sensor network, a large number of camera nodes form a distributed system,

where the camera nodes are able to process image data locally and to extract relevant information, to collaborate with other cameras on the application-specific task, and to provide the system's user with information-rich descriptions of captured events. With current trends moving toward development of distributed processing systems and with an increasing number of devices with built-in image sensors, a question of how these devices can be used together appears [75].

### 2.2.2 Challenges in VSNs

Visual sensor networks are in many ways unique and more challenging compared to other types of wireless sensor networks. These differences add another dimension to existing problems in wireless sensor networks. For instance, most sensors provide measurements as temporal signals that represent physical quantities such as temperature. On the other hand, at each time instant image sensors provide a 2D set of data points, which we see as an image. This results in richer information content as well as in a higher complexity of data processing and analysis.

As in other types of sensor networks, requirements of resources such as energy and bandwidth forms one of the main problems in visual sensor networks. The constraints in hardware architecture (power of local processors, availability of larger memories, etc.) affects the complexity of the algorithms that can be run on cameras. More complex algorithms output higher level information which reduces the amount of data to be transmitted. In most of the applications, data flow is required to be in real-time. In one hand, real-time performance of visual sensor networks is related to power of local processors. On the other hand, it is related to transmission of the processed data throughout the network which is constrained by the wireless channel limita-

tions (available bandwidth, modulation, data rate), employed wireless standard, and by the current network condition. Time synchronization may become important for tasks that involve multiple cameras. Camera locations are required for most of the multi-camera image processing algorithms. It is also important to know the vision graph of the network to decide efficient communication rules. In addition, calibrated cameras are needed in some of the multi-camera algorithms. In a network of cameras constrained by limited resources, camera collaboration should be managed intelligently.

We believe that camera calibration and time synchronization problems are not related to the scope of this thesis. In our experiments, we assume that all cameras in VSNs are already calibrated and synchronized when our approach is used. Sensor selection and sensor resource management (sensor tasking) are also out of scope of this thesis. We have assumed that one of the camera nodes, relatively more powerful one, has been selected as the fusion node. In the next subsections the problems which are in the scope of this thesis are explained in detail.

**Energy & Bandwidth Constraints**

The lifetime of battery-operated camera nodes is limited by their energy consumption, which is proportional to the energy required for sensing, processing, and transmitting the data. Given the large amount of data generated by the camera nodes, both processing and transmitting image data are quite costly in terms of energy, much more so than for other types of sensor networks. Furthermore, visual sensor networks require large bandwidth for transmitting image data. Thus both energy and bandwidth are even more constrained than in other types of wireless sensor networks.

In visual sensor networks, the processor in camera nodes give us the ability to process images directly on the camera, thereby remove the requirements of collecting images in a central processor and decrease communication needs. In order not to ruin this, the feature extraction algorithms that run on the processors should not be heavy-weight. Computationally, they should be light, but they are supposed to be robust as well. VSNs provides us the opportunity of having more than one cameras to cover an area, thereby providing us much more information about the scene. But this opportunity may create bottlenecks in communication while fusing the multi-view information. For this reason, the features that will be shared and fused should be small in size.

Following the points mentioned above, the feature extraction algorithms that run on camera nodes are supposed to be light-weight, robust and the resulting features should be small in size.

### Collaboration Between Cameras

Visual sensor networks are envisioned as distributed and autonomous systems, where cameras collaborate and, based on exchanged information, reason autonomously about the captured event and decide how to proceed. Through collaboration, the cameras relate the events captured in the images, and they enhance their understanding of the environment.

As we have pointed before, the collaboration of cameras provides richer information about the scene and it may also create a burden in the communication of the network.

Therefore, the approach for running inference on the information shared among cameras is very important. This collaborative inference should include all the cameras without overloading the network. Since collecting all the information in a single node will create bottlenecks in the communication, the inference should not be centralized.

In the next section, we have given more details about how inference should be run by going through the methods proposed for surveillance applications.

### 2.2.3 Building a Surveillance System in VSNs

Surveillance is the main application field for VSNs in this thesis. Thus, in this section, we discuss about human tracking and action recognition methods. First, we describe the estimation algorithms that are proposed for performing collaborative processing. Then, we give examples of such algorithms applied for human tracking and action recognition tasks in VSNs.

Let us consider a network $C = \{C_1, \cdots, C_c, \cdots, C_N\}$ of $N$ cameras monitoring $T$ targets. Let the state of target $i$ at time $k$ be defined as $x_k^i$. The elements composing the state depend on the application. For tracking, in the simplest case, the state is defined by the position of the target. In more complex cases, the state contains other elements such as, for example, shape and velocity parameters. For action recognition, the state describes the action performed by $i$th person. The goal is to estimate the target state $x_k^i$ by fusing the data gathered from the cameras. In tracking, target state estimation aims to associate noisy measurements $Z_k^i = \{z_1^i, \cdots, z_k^i\}$ belonging to the same person over time to obtain the trajectory $X_k^i = \{x_1^i, \cdots, x_n^i\}$ for each person $i$. In action recognition, the activity of person $i$, $x_k^i$, is obtained by classifying

the features extracted from the measurements from the cameras, $y^i = \{y_1^i, \cdots, y_N^i\}$.

In both tracking and action recognition applications, the amount and type of information sharing for state estimation differs from method to method. Depending on the algorithm, type of cameras in the network, and on the strategy adopted for data fusion, multicamera approaches incur different computation and communication costs. Moreover, the data-gathering (fusion) strategy has a significant influence on the scalability of the network and on the communication cost, thus affecting the applicability of a method. Multicamera approaches can be categorized, based on intersensor communication, into three main groups [76], centralized, decentralized, and distributed estimation.

Centralized estimation is performed in a single node that receives (raw or processed) data from each camera in the network. Although centralized approaches can exploit directly existing single-camera trackers/action classifiers on the fused data [77, 4], the presence of a single global fusion center leads to high data-transfer rates and to a lack of scalability and energy efficiency.

In decentralized estimation, cameras are grouped into clusters and member nodes only communicate with their local fusion centers, which are nodes in a network that collect data from the cameras within a cluster and perform state estimation [78, 79] (Figure 2.8). The use of fusion nodes favors scalability and reduces the overall communication load by limiting the flow of measurements from nearby nodes within a cluster and among fusion nodes. Features are extracted in each camera view and then projected to the fusion nodes for multi-view estimation. Finally, fusion nodes

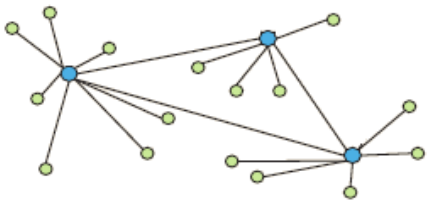


**Figure 2.8**: In decentralized approaches cameras are grouped into clusters.

communicate with each other to handoff tracking/action recognition tasks over the network [7]. Energy efficiency can be improved by selectively activating only cluster members that are expected to observe the targets of interest.

Fusion nodes can be chosen a priori (fixed fusion nodes) or dynamically. Fixed fusion nodes are generally used in networks where some nodes have higher processing power and energy supply [78]. Although fixed fusion nodes reduce the computational cost for cluster members and increase the lifespan of the overall network, in some cases, they do not necessarily use the cameras with the best view of a target and, hence, may generate lower-quality observations. To compensate for this limitation, fusion nodes can be chosen dynamically based on trackability measures that evaluate the effectiveness of the features observed by a camera [80]. However, dynamic fusion nodes can select the best view in a cluster, which may not necessarily be the best cameras for tracking among all cameras. There might be cameras belonging to another cluster that can better observe the target. Since the cluster members are fixed and the tracking task has not yet been handed off to their cluster, these cameras may not be used. To avoid such cases, camera clustering should be adapted online, and cameras should be added to and removed from the clusters based on target observability [7].

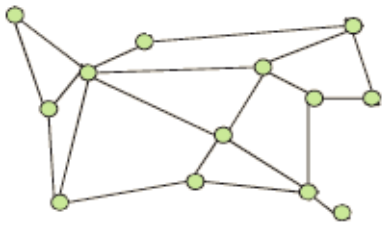


**Figure 2.9**: In distributed approaches there are no local fusion centers.

To further increase scalability and to reduce communication costs, distributed estimation operates without local fusion centers. The estimates generated in a camera are transmitted to its immediate neighbors only (Figure 2.9). The received estimates are used to refine the next-camera estimates, and these refined estimates are then transmitted to the next neighbor [81, 82, 9, 17]. This process is completed after a predefined number of steps after all cameras viewing the target are visited or when the uncertainty has decreased below a desired value.

Other than the approaches classified in three groups above, there has been some simple work that uses basic features or techniques to adapt centralized approaches to VSNs. For instance, visual hulls are used in [83, 84] to detect the presence and number of humans. But, since visual hull presents the largest volume in which a human can reside, the exact number of humans cannot be determined when humans come closer. To minimize the amount of data to be communicated between cameras, in some methods simple features are used for communication. For instance, 2D trajectories are used in [12]. In [13], 3D trajectories together with color histograms are used. Hue histograms along with 2D position are used in [14].

In the following subsections, decentralized and distributed approaches proposed to

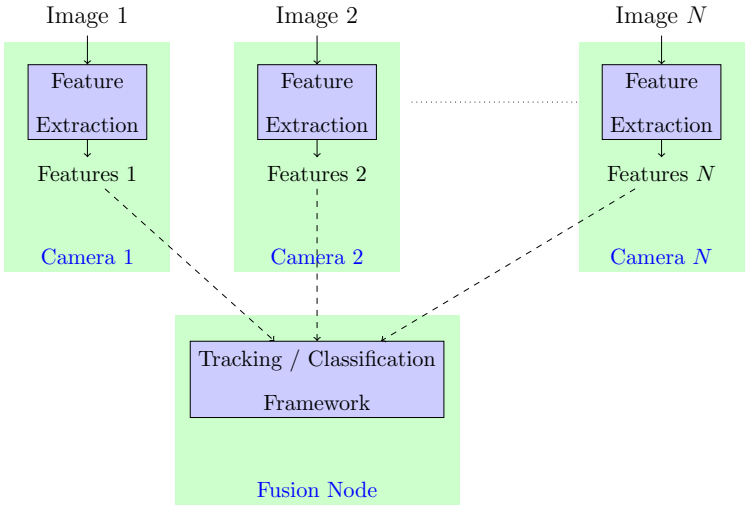



**Figure 2.10**: The flow diagram of decentralized approaches.

perform tracking/action recognition tasks in VSNs are discussed in detail.

### Decentralized Approaches

Traditional trackers/action classifiers can be extended to exploit multicamera data and used in fusion nodes (cluster heads), which receive raw or filtered data from the cameras in a cluster (Figure 2.10). Decentralized approaches can be implemented easily by using existing centralized approaches in a fusion node.

These kind of approaches have been applied to multi-camera target tracking problem in various ways [15, 7, 16]. For a nonoverlapping camera setup, tracking is performed by matching the people in the scene with the entry/exit zones in camera views. This is done by maximizing the similarity between the observed features from each camera and minimizing the long-term variation in appearance using graph matching at the fusion node [15]. They pose the problem of tracking in a camera network as a multi-

objective optimization problem. Based on the centralized approach in [85], the first objective is to choose the tracks such that they maximize the similarity between the observed features in a local neighborhood. To make the method robust to changes in lighting and pose, scene clutter and occlusions, they introduce a second optimization objective that minimizes the long-term variation in appearance and identity of the tracked objects over the entire camera network. When a person disappears or a new person enters, the features are sent to a fusion node to perform integration over space and time between cameras. Since the setup in this work is nonoverlapping cameras, tracking is performed by a single camera that observe the person. For this reason, the data fusion that will burden the communication in the network occasionally happens. Thus, it is not clear whether this method will decrease the communication when there is a overlapping camera setup in the network.

For an overlapping camera setup, a cluster-based Kalman filter in a network of wireless cameras is proposed in [7, 16]. Local measurements of the target acquired by members of the cluster are sent to the fusion node. Then, fusion node estimates the target position via an extended Kalman filter, relating the measurements acquired by the cameras to the actual position of the target by nonlinear transformations. In [7], when a target is detected, cameras that can observe the same target interact with one another to form a cluster and elect a cluster head. Local measurements of the target acquired by members of the cluster are sent to the cluster head, which then estimates the target position via Kalman filtering. The underlying clustering protocol of their earlier work [86] allows the current state and uncertainty of the target position to be easily handed off among clusters as the person is being tracked. In the experiments, the method is applied on a network of cameras that are mounted in the ceiling of

their laboratory. In this camera setup, person detection is an easy task compared to detection problem in standard camera setups and avoid noisy/uncertain observations. For this reason this makes the experiments unrealistic for many surveillance applications. In [16], by using the same clustering protocol face tracking is performed by Kalman filtering.

A user-programmable framework that processes multi-camera data along with prior context information to infer activities is proposed in [87]. Their approach does not look at body gestures but at more macroscopic (higher level) activities conducted in the context of a building or a city map. Instead of applying complex image-processing techniques to maximize the information extracted from each camera sensor node, they use lightweight algorithms [88] to detect a person's position with respect to a set of predefined areas in a building. In order to translate raw sensing data to high-level interpretations, such as human activities, they use probabilistic context-free grammars (PCFGs), which is a syntactic pattern recognition technique, organized in hierarchies to classify spatial and temporal patterns from simple low-level sensor measurements. The inference on PCFGs is run on the fusion node, but the bandwidth requirements of the approach are not specified. In the experiments, as it is done in the previous method, a network of cameras mounted in the ceiling, which is an unnatural situation for many surveillance applications, are used.

A distributed and lightweight activity classification algorithm is proposed in [89]. At each camera, similar to [90] multi-view spatio-temporal histogram features are extracted and test histograms are transmitted to all cameras. Then, each camera in the network calculates the distance between the multi-view test histograms and

multi-view training histograms and transmits the distance values to a fusion node in order to find the label of the performed action.

A method, focused more on human body parts, for human pose estimation is proposed in [91]. Inspired by the concepts from the top-down and bottom-up approaches, they propose a layered analysis for human pose estimation. The bottom-up approach fits well to the requirement of local processing, since image data are locally reduced to body part descriptors or image features that are affordable for network transmission. The body model of the top-down approach embodies up-to-date information from both current and historical observations of all cameras in a concise way. They perform feature extraction such as color-based segmentation and ellipse fitting to body parts in each camera view. The shape fitting results are transmitted to the fusion node to construct the pose.

In [92], a multi-view human tracking algorithm that is based on collaborative signal processing mechanisms is proposed. At each camera node, foreground of the scene is detected using the approach of [93] and human classification is performed to locate the people on the foreground [94]. The observations from each node are represented as Gaussian distributions and fused from one node to another in a distributed way to obtain a multi-view likelihood function. Once a node receives observations from other nodes, it integrates these with its own observations and searches for an optimal node by considering the communication costs (link bandwidth, transmission latency, etc.), location, direction and observation accuracy of nodes. When the entropy of fused likelihood functions exceeds a threshold, the last node sends the final likelihood function to the fusion node. In the fusion node, tracking is performed using

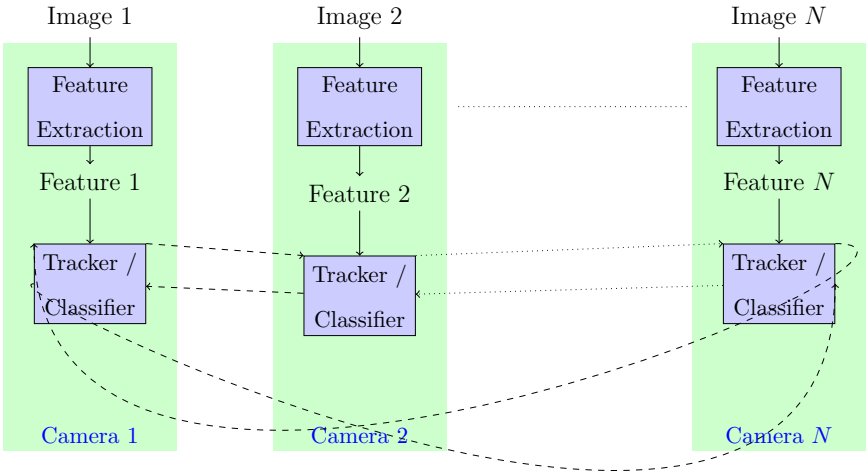



**Figure 2.11**: The flow diagram of distributed approaches.

a particle-filter that obtains a transition model from radial basis functions trained with previous estimation results.

### Distributed Approaches

Unlike decentralized approaches, distributed trackers/action classifiers have no specific local fusion centers: each node fuses its estimates with information received from its neighbors and projects the updated estimates to the next neighbor until the last node is reached or a desired accuracy is achieved (Figure 2.11).

In [9], by leveraging upon concepts of consensus that have been studied in the context of multiagent systems, a distributed scene analysis method that is focused on track-

ing and action recognition is proposed. Each camera estimates certain parameters based upon its own sensed data which is then shared locally with the neighboring cameras in an iterative fashion, and a final estimate is arrived at in the network using consensus algorithms. For multitarget tracking in a distributed camera network, they show how the Kalman-Consensus algorithm [95] can be adapted to take into account the directional nature of video sensors and the network topology. For the activity recognition problem, they derive a probabilistic consensus scheme that combines the similarity scores of neighboring cameras to come up with a probability for each action at the network level.

A wireless embedded smart camera system for cooperative human tracking and detection of composite, semantically high-level and user-defined events is proposed in [17]. At each camera lightweight and robust foreground detection [96] and color histogram based tracking [97] algorithms are implemented and run on the microprocessor of the camera board. To solve data association problem, the field of view lines of cameras are used to check if a person is being tracked by another camera and a label for the person is requested to achieve consistent labeling. To address limited energy, limited memory and bandwidth issues, every frame or trajectory is not saved or transferred. Important portions of video and trajectories are determined by detecting events of interest that are pre-defined beforehand by users. Three primitive events are defined: motion detection, occurs when motion is detected in a region of interest; tripwire crossing, occurs if a person crosses a specified line in the specified direction; abandoned object occurs in a specified region when there is an object abandoned for more than a specified time. More complex scenarios are also defined by sequencing the primitive events. Communication in the network is minimized by sending messages

only when an event of interest occured or to assign labels to tracked people. But, how action recognition is performed in a distributed way is not clear.

# 3 ACTION RECOGNITION USING SPARSE REPRESENTATION

As we have seen in Section 2.1.4, in some applications sparse representation has been used as a classification method. In this section, we present our novel multi-camera action recognition method that is based on sparse representation [98]. This method is one of the contributions of this thesis. To the best of our knowledge, this is the first action recognition method that uses sparse representation. As in [11, 99], we assume that a test sample can be written as a linear combination of training samples from the class it belongs to. In other words, a test sample has a sparse representation in the space covered by the training samples. Based on this assumption, we cast the classification problem as an optimization problem and solve it by enforcing sparsity through $l_1$ regularization. We develop two parallel perspectives one based on regular sparsity and the other one based on so called group sparsity. The sparse approach is based on the idea that a test sample can be represented by a small number of training samples, regardless of the class labels of the training samples. On the other hand, the group sparse approach imposes more structure, imposing sparsity across classes (i.e., allowing only a small number of classes to be active in the representation) while allowing the use of a large number of training samples from the active classes. In the experimental results, we demonstrate the superiority of our method, especially when

observations are low resolution, occluded, and noisy and when the feature dimension is reduced. We present our action recognition method and experimental results in Section 3.1.

Since our action recognition method is based on a centralized approach and the data (feature) transmission in the network requires high bandwidth, application of such a system in VSNs will be problematic. In Section 3.2, we describe our proposed solution to overcome this problem. Recently, the role of sparsity in classification has been questioned and for the face recognition problem, it has been shown that the $l_1$-norm constraint may not be necessary [18, 19]. Based on these studies, we have analyzed this issue for two different action recognition problems and obtained similar observations showing that optimization based on $l_2$ regularization can also achieve a level of accuracy close to $l_1$ based regularization. Our analysis and observations are presented in Section 3.3.

## 3.1 Classification via Sparse Representation

Subspace modelling is a well-known approach applied in many classification problems such as face recognition [67, 63, 70], object recognition [100] and facial expression recognition [101]. Similarly, in the feature space, we assume that each action class satisfies a low–dimensional subspace model. If a valid test sample can be represented as a linear combination of all training samples, the dominant coefficients in the sparsest representation correspond to the training samples from the underlying action class, and hence they indicate the membership of the test sample.

As we have seen in Section 2.1.4, mathematically we express this as follows: in a feature space of $m$ dimensions, given $N_i$ training samples of the $i$th action class, there is a relation between a test sample, $y \in \mathrm{R}^m$, and the training samples, $\{v_{i,j}\}_{j=1}^{N_i}$, from the same class:

$$y = \alpha_{i,1}v_{i,1} + \alpha_{i,2}v_{i,2} + \cdots + \alpha_{i,n_i}v_{i,N_i} \tag{3.1}$$

where $\alpha_{i,j} \in \mathrm{R}, j = 1, 2, \cdots, N_i$.

Writing the training samples of $i$th class as the columns of a matrix, we obtain the matrix $\psi_i = [v_{i,1}, v_{i,2}, \cdots, v_{i,N_i}] \in \mathrm{R}^{m \times N_i}$. Since we do not know the class of the test sample initially, by concatenating the $N$ training samples of all $k$ object classes, we obtain the following matrix $\psi$:

$$\psi \doteq [\psi_1, \psi_2, \cdots, \psi_k] \in \mathrm{R}^{m \times N} \qquad N = \Sigma_{i=1}^{k} N_i \tag{3.2}$$

As in [11, 99], by rewriting the relation in Eq. 3.1 using the matrix $\psi$, we obtain the following linear representation of $y$ in terms of all training samples:

$$y = \psi x \in \mathrm{R}^m \tag{3.3}$$

where $x = [0, \cdots, 0, \alpha_{i,1}, \alpha_{i,2}, \cdots, \alpha_{i,n_i}, 0, \cdots, 0]^T \in \mathrm{R}^N$ is a coefficient vector, all of whose entries except those corresponding to the $i$th class are zero. Thus, solving the system in Eq. 3.3 for $x$ gives the identity of the test sample $y$. In practice, since real data are noisy, it may not be possible to express the test sample exactly as a superposition of the training samples. In other words, a noise term can be added to the linear system in Eq. 3.3:

$$y = \psi x + z \tag{3.4}$$

where $z \in \mathrm{R}^m$ is a noise term with bounded energy $||z||_2 < \epsilon$.

For a large number of object classes, this representation is naturally *group sparse* – meaning that there are non-zero coefficients corresponding to a particular group (class) of training samples and zeros elsewhere. Different from the procedure in [11, 99], we define a vector, $x'$, as follows:

$$x' = [x'_1, x'_2, \cdots, x'_k] \in \mathrm{R}^k \quad x'_i = ||\{\alpha_{i,j}\}_{j=1}^{N_i}||_2 \tag{3.5}$$

Namely, the $i$th element of the vector $x'$ is the $l_2$ norm of the coefficients corresponding to training samples of the $i$th class. Since the vector $x$ is group sparse, $x'$ is a sparse vector. Therefore, to classify the test sample, we are interested in finding the group sparse solution to $y = \psi x + z$ by solving the following optimization problem:

$$\hat{x} = \arg\min_x ||x'||_0 \text{ subject to } ||y - \psi x||_2 < \epsilon \tag{3.6}$$

where $||.||_0$ denotes the $l_0$ norm and counts the number of nonzero entries in a vector. However, the problem of minimizing the $l_0$ norm is NP-hard. Recent development in the emerging theory of compressed sensing [29, 65, 66] reveals that if the vector $x'$ is sparse enough, the solution of the $l_0$-minimization problem in Eq. 3.6 is equal to the solution to the following $l_1$-minimization problem:

$$\hat{x} = \arg\min_x ||x'||_1 \text{ subject to } ||y - \psi x||_2 < \epsilon \tag{3.7}$$

In fact, we optimize the Lagrangian form of this problem:

$$\hat{x} = \arg\min_x ||y - \psi x||_2 + \lambda ||x'||_1 \tag{3.8}$$

where $\lambda$ is the regularization parameter. This optimization problem is convex [102, 103, 104] and it can be efficiently solved via second-order cone programming [32]

After finding the group sparse representation $\hat{x}$ of the test sample $y$, we classify it based on how well the coefficients associated with all training samples of each action class reproduce $y$. For each class $i$, let $\delta_i : \mathrm{R}^n \rightarrow \mathrm{R}^n$ be the characteristic function that selects the coefficients associated with the $i$th class. For $x \in \mathrm{R}^n$, $\delta_i(x) \in \mathrm{R}^n$ is a new vector whose only nonzero entries are the entries in $x$ that are associated with class $i$. Using only the coefficients associated with the $i$th class, one can approximate the given test sample $y$ as $\tilde{y} = \psi\delta_i(\hat{x}_i)$. We then classify $y$ based on these approximations by assigning it to the object class that minimizes the residual between $y$ and $\tilde{y}$:

$$\hat{i} = \arg\min_i r_i(y) \doteq \arg\min_i ||y - \psi\delta_i(\hat{x}_i)||_2 \tag{3.9}$$

In [11, 99], it is assumed that the test sample can be represented by a small number of training samples from the same class and hence the vector $x$ is considered as sparse (rather than group sparse) (Section 2.1.4). A similar formulation could be obtained in our setting by replacing $x'$ by $x$ in Eqs. 3.7 and 3.8. After finding the sparse representation, the test sample can be classified again by using Eq. 3.9. We also present the results of this sparsity-based approach and compare it to the group sparsity based approach described above.

### 3.1.1 MHVs and Action Descriptors

Motion history volumes are extensions of 2-D motion templates, first introduced by Bobick and Davis in [105], to 3-D [4]. They represent the dynamics of the motion in 3-D.

At each camera, after acquiring the images, silhouettes are extracted. The silhouette images obtained from multiple cameras are used to create visual hulls at each time instance. By using these visual hulls an occupancy function, $D(x, y, z, t)$, that represents the presence of a person in space and time, is defined. $D(x, y, z, t)$, is set to 1 if the point $(x, y, z)$ is 1 in the visual hull created at time $t$, and set to 0 otherwise. By using this occupancy function, a motion history volume is constructed as follows:

$$v_\tau(x, y, z, t) =$$

$$\begin{cases} \tau & \text{if } D(x, y, z, t) = 1 \\ max(0, v_\tau(x, y, z, t-1) - 1) & o.w. \end{cases} \tag{3.10}$$

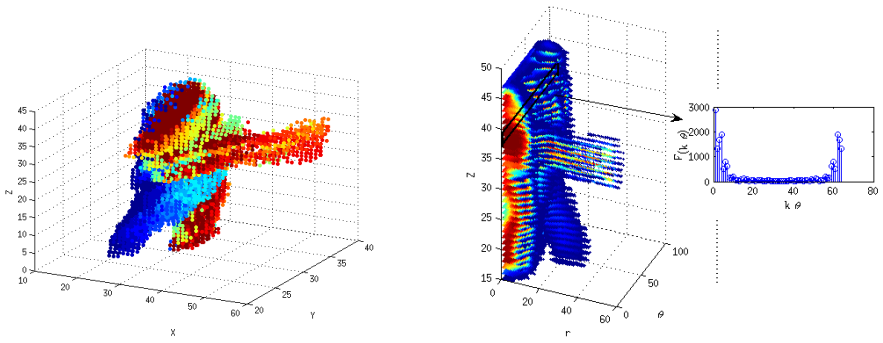
where $\tau$ is the maximum duration of the motion at point $(x, y, z)$ [4].

With respect to the duration of an action, the volumes found by Eq. 3.10 are normalized and final motion history volumes are obtained:

$$v(x, y, z) = \frac{v_{\tau=t_{max}-t_{min}}(x, y, z, t_{max})}{t_{max} - t_{min}} \tag{3.11}$$

where $t_{min}$ and $t_{max}$ are start and end time of an action. $t_{min}$ and $t_{max}$ are estimated by searching for the local minima in the global motion energy of MHVs [4]. An example of MHV constructed for "kicking" action is shown in Figure 3.1-a. To be able to recognize actions robustly, a method that is invariant to rotation, scale and translation is needed. But, since MHVs encode space occupancy, they are not invariant. Because of the nature of human motions, it is reasonable to assume that similar actions only differ by rigid transformations composed of scale, translation, and rotation around the z-axis [4]. We express MHVs in a cylindrical coordinate-system:

$$v(\sqrt{x^2 + y^2}, tan^{-1}\left(\frac{y}{x}\right), z) \rightarrow v(r, \theta, z) \tag{3.12}$$



(a) (b)

**Figure 3.1**: (a) An example of MHV constructed for "kicking" action. Color indicates the values of MHV. (b) Action descriptors are constructed by taking Fourier transform over $\theta$ for couples of values $(r, z)$ in cylindrical coordinates and concatenating the Fourier magnitudes.

Thus rotations around the z-axis results in cyclical translation shifts:

$$v(x \cdot cos\theta_0 + y \cdot sin\theta_0, -x \cdot sin\theta_0 + y \cdot cos\theta_0, z) \rightarrow v(r, \theta + \theta_0, z) \tag{3.13}$$

The absolute values of 1-D Fourier transform along the $\theta$ dimension for each value of $r$ and $z$, $|V(r, k_\theta, z)|$, are used as motion descriptors:

$$V(r, k_\theta, z) = \int_{-\pi}^{\pi} v(r, \theta, z) e^{-j2\pi k_\theta \theta} d\theta \tag{3.14}$$

Motion descriptor extraction for "kicking" is illustrated in Figure 3.1-b.

By the *shift property* of Fourier transform, a shift in the $\theta$ dimension corresponds to phase modulation in frequency domain. As a result, 1-D Fourier magnitudes are invariant to rotation along $\theta$. Before taking the Fourier transform, the location and scale dependencies of MHVs are removed by centering around the center of mass, and scale normalization. Therefore, the motion descriptors obtained by this procedure

**Figure 3.2**: Example views from the IXMAS dataset recorded by five synchronized and calibrated cameras [4].

are invariant to rotation, scale and translation. We use these descriptors as features in our method.

### 3.1.2 Experimental Results

#### Setup

We test our method on the publicly available IXMAS dataset [4], which is a popular dataset used for evaluating multi-view action recognition methods. The dataset consists of 11 actions (check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave, punch, kick, point) and each action is performed three times with free orientation and position by 10 different actors. Actions are recorded by five synchronized and calibrated cameras. Example views from the dataset are shown in Figure 3.2. We use the visual hulls provided with the dataset on a 64x64x64 voxel grid. While constructing the MHVs, the motion segmentation method in [4] is used. Action descriptors are created on a 32x32x32 voxel grid. The classification results presented here are based on leave-one-out cross-validation, i.e., we train on data from 9 actors and test on the remaining actor; repeat this for all combinations of actors; and average the results.

We have compared our method with the method in [4]. In [4], first principal compo-

nent analysis (PCA) is applied for dimensionality reduction and then, three different procedures are performed to classify action descriptors: 1) a new action is classified according to the Euclidean distance to class means; 2) a new action is classified according to the Mahalanobis distance to class means; 3) Fisher linear discriminant analysis (LDA) is performed to maximize the between-class scatter and minimize the within-class scatter, then a new action is classified according to the Euclidean distance to class means.

**Comparison of $l_1$-minimization Solvers**

In Section 2.1.2, we have seen that solving optimization problems with $l_1$ constraint has become a well-established research area. There are many solver algorithms that has been proposed for $l_1$-minimization. In order to choose a solver that is fast and accurate for our sparse representation based classification framework, we have compared the solvers described in Section 2.1.2.

In the comparison, we have used the experimental setup that we have explained in previous section. PDIPA(CVX) [106], Homotopy [107], L1LS [38], SpaRSA [47], FISTA [108] and ALM [109] algorithms have been used to solve the optimization problem in Eq. 3.8 using the *sparse* approach (replacing $x'$ by $x$). The regularization parameter in Eq. 3.8, $\lambda$, is set to 1. The average run-times of the solvers in seconds and average accuracy rates obtained after classifying test samples using the sparse solution has been shown in Table 3.1. To avoid trivial solutions, we have also checked the number of iterations of each algorithm . The average iteration count solvers are also given in Table 3.1.

| | Accuracy | Run-time (sec.) | Iteration Count |
|---|---|---|---|
| PDIPA(CVX) [106] | 92.42% | 16.2551 | 11.3152 |
| Homotopy [107] | 79.39% | 0.2990 | 33.0061 |
| L1_LS [38] | 92.42% | 8.7521 | 10.2061 |
| SpaRSA [47] | 8.79% | 4.3625 | 566.6940 |
| FISTA [108] | 55.45% | 0.5398 | 13.7848 |
| ALM [109] | 17.87% | 0.0210 | 2 |

**Table 3.1**: Average run-times, iteration counts of solvers and accuracy rates for action recognition.

We can see that ALM, FISTA and Homotopy algorithms are the fastest solvers. But, ALM and FISTA could not achieve good performance in classification. Especially for ALM, average iteration counts show that it finds the trivial solution. When we look at the Homotopy algorithm, we can see that, it works fast, but obtains a reasonable level of accuracy. On the other hand, both CVX and L1LS algorithms achieves high level of accuracy in reasonable run-times. Since the current version of L1LS algorithm is implemented to solve only the sparse approach in Section 3.1, it does not allow us to define $x'$ variable in Eq. 3.8 and solve the group sparse approach. Based on these observations, we select the CVX algorithm for our action recognition experiments.

### Action Recognition Results

Table 3.2 presents the performance of our method and the method in [4] for each action and averaged over all actions. For the framework proposed in this paper, we presented the results of both the *group sparse* approach based on Eq. 3.8 as well as the *sparse* approach. We have empirically set the regularization parameter in Eq. 3.8, $\lambda$, to 500 and 100 for the group sparse and sparse approaches, respectively. For the results of [4] in Table 3.2, the column titled "PCA" corresponds to

| | **The method in [4]** | | | **SR** | |
|---|---|---|---|---|---|
| Action | LDA | PCA | Maha. | Group Sparse | Sparse |
| Check watch | 83.33% | 46.66% | **86.66%** | 80.00% | 83.33% |
| Cross arms | **100.00%** | 83.33% | **100.00%** | **100.00%** | **100.00%** |
| Scratch head | 93.33% | 46.66% | 93.33% | **96.66%** | **96.66%** |
| Sit down | 93.33% | 93.33% | 93.33% | **96.66%** | **96.66%** |
| Get up | **90.00%** | 83.33% | 93.33% | **90.00%** | **90.00%** |
| Turn around | **96.66%** | 93.33% | **96.66%** | **96.66%** | **96.66%** |
| Walk | **100.00%** | **100.00%** | **100.00%** | **100.00%** | **100.00%** |
| Wave hand | **90.00%** | 53.33% | 80.00% | 83.33% | 86.66% |
| Punch | 93.33% | 53.33% | 96.66% | **96.66%** | **96.66%** |
| Kick | 93.33% | 83.33% | **96.66%** | **96.66%** | **96.66%** |
| Pick up | 83.33% | 66.66% | **90.00%** | **90.00%** | **90.00%** |
| Average | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** |

**Table 3.2**: Accuracies of the method in [4] and our SR based method for each action. Bold values represent the best accuracy for each action.

the Euclidean distance-based approach in [4] and the other two columns correspond to the LDA and Mahalanobis distance-based versions. It can be seen that in three actions our method achieves a better level of accuracy than the method in [4]. In six actions, our method and the method in [4] achieves the same results. Just for "check watch" and "wave hand" actions, the method in [4] achieves better results. In the average, group sparse version of our method achieves the best level of accuracy.

We have also performed tests under various conditions. In the next subsections, the results of experiments when action descriptors are low resolution, when data are noisy, when there is occlusion are presented.

**Low Resolution Data**

In this experiment, action descriptors created in lower voxel grid sizes are used to test the robustness of our method in the case of resolution loss. Our method and the method in [4] are tested by using action descriptors in 16x16x16 and 8x8x8 voxel grid sizes. The average accuracies obtained in these experiments together with the average accuracies obtained using the 32x32x32 action descriptors are presented in Table 3.3.

These results show that even when the action descriptors have very low resolution our method achieves reasonable level of accuracy. For the 16x16x16 grid size, the performance of the method in [4] degrades much more dramatically than that of our method. While the method in [4] achieves reasonable level of accuracy (78.18%), our method achieves better accuracy level (90.00%). When the action descriptors have a resolution of 8x8x8, the method in [4] exhibits an unacceptable level of accuracy. PCA and Mahalanobis procedures achieve only random assignment accuracies (9.09%), whereas our method achieves a reasonable level of accuracy (73.64%). These experiments demonstrate the robustness and superiority of our proposed approach in the case of low resolution data.

In Table 3.3, we have also presented the results of the *sparse* version of our approach. When the action descriptors have a resolution of 16x16x16, sparse version performs slightly better than the group sparse version (90.61%). But for the resolution of 8x8x8, the sparse version achieves a lower level of accuracy than the group sparse version (67.88%).

| | The method in [4] | | | SR | |
|---|---|---|---|---|---|
| $y$ | LDA | PCA | Maha. | Group Sparse | Sparse |
| [32] | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** |
| [16] | 78.18% | 44.55% | 77.27% | 90.00% | **90.61%** |
| [8] | 20.91% | 9.09% | 9.09% | **73.64%** | 67.88% |

**Table 3.3**: Average accuracies of the method in [4] and our SR based method when action descriptors are low-resolution. Bold values represent the best accuracy for each row.

### Corrupted Data

Poor performance in temporal segmentation affects the values of the MHVs. If the start and/or end times of the motion is miscalculated, the values of MHVs will be inaccurate (Eq. 3.11). To test the robustness of our method for such perturbations, we have corrupted the MHVs with zero-mean Gaussian noise. The test sample is created by extracting the action descriptors from these corrupted MHVs. Training samples are created by using the original MHVs. We have performed experiments with various noise variances which are specified in terms of percentages of the maximum values of the MHVs.

Average accuracies achieved by the method in [4] and our method for various noise levels are presented in Table 3.4 and Figure 3.3. The results obtained from the original data (0% corruption) are also shown for comparison. It can be observed that our method outperforms the method in [4] for all noise variances considered in this experiment. The lowest accuracy achieved by our method is when the noise variance is 90% of the maximum of the MHV and it is a reasonable rate (81.82%). For all variances, we also observe that group sparse and sparse versions of our approach

| | | **The method in [4]** | | | **SR** | |
|---|---|---|---|---|---|---|
| $y$ | Corruption | LDA | PCA | Maha. | Group Sparse | Sparse |
| [32] | 0% | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** |
| [32] | 10% | 89.39% | 42.42% | 90.30% | 92.73% | **93.33%** |
| [32] | 20% | 65.15% | 16.06% | 63.03% | **93.03%** | 92.73% |
| [32] | 30% | 25.76% | 9.09% | 24.55% | **91.52%** | 90.61% |
| [32] | 40% | 11.52% | 9.09% | 13.33% | **92.42%** | **92.42%** |
| [32] | 50% | 11.21% | 9.09% | 12.42% | **90.91%** | 90.30% |
| [32] | 60% | 10.00% | 9.09% | 10.30% | 89.39% | **90.30%** |
| [32] | 70% | 10.61% | 9.09% | 10.00% | **86.67%** | 86.36% |
| [32] | 80% | 9.70% | 9.09% | 9.39% | **83.94%** | **83.94%** |
| [32] | 90% | 9.70% | 9.09% | 9.09% | **81.82%** | 80.91% |
| [32] | 100% | 9.39% | 9.09% | 9.09% | **84.55%** | 83.94% |

**Table 3.4**: Average accuracies of the method in [4] and our method on data corrupted by zero-mean Gaussian noise with variance specified in terms of percentages of the maximum value of the MHV. Bold values represent the best accuracy for each row.

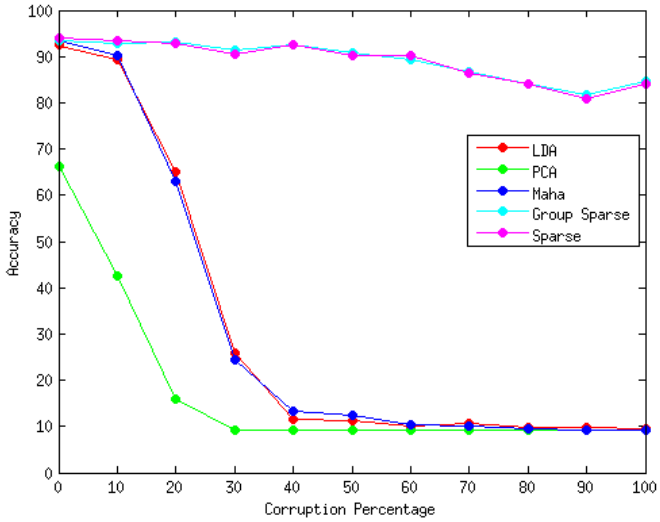



**Figure 3.3**: Accuracies of the method in [4] and our method on data corrupted by zero-mean Gaussian noise with variance specified in terms of percentages of the maximum value of the MHV.

achieve similar results. On the other hand, there is a significant performance drop for the method in [4] as the data become more noisy (most notably when the noise variance increases from 20% to 30%). For the tests in which the noise variance is selected equal to or greater than 60% of the maximum of the MHV, the best accuracy obtained by the method in [4] is close to random assignment accuracy (9.09%). These results show that our method is robust to reductions in data quality, which may be the result of, e.g., failures in temporal segmentation.

### Occluded Data

Occlusion is one of the most common and important problems in real world scenarios. Occlusion occurs most commonly through an object occluding parts of the person in the scene. Since the visual hulls are constructed by using these occluded silhouettes,

they will also be occluded and, consequently, MHVs will be occluded as well.

In this experiment, we have examined how the occlusion in MHVs affects the recognition accuracy. Starting from the center of the MHV, we have occluded (set to zero) the MHVs in various levels, from 5 percent to 90 percent. Test samples are created by extracting the action descriptors from these occluded MHVs and training samples are created by using the original MHVs. Due to the steps involved in feature extraction, occlusion of MHVs has a non-trivial effect in the feature space [4]. In particular, this effect involves all feature components rather than being limited to occlusion of a subset of the feature components. Given this non-trivial effect, in all of our experiments with all techniques, we assume the presence of a perfect occlusion detector for the sake of simplicity. The occluded points have not been taken into account in feature extraction steps of both our method and the method in [4].

In Table 3.5 and Figure 3.4, the average accuracies obtained by the method in [4] and our method for various levels of occlusion are presented. The results obtained from the original data (0% occlusion) are also presented for comparison. The results show that our method performs better than the method in [4] for all levels of occlusion. For occlusion levels up to 60%, the accuracies of the method in [4] are close to the accuracies of our method. But, for higher occlusion levels, our method is definitely better than the method in [4].

The accuracies of the *sparse* version are also presented in Table 3.5. Comparing the results of the group sparse and sparse versions, it can be observed that the two versions achieve similar accuracy levels.

| | | The method in [4] | | | SR | |
|---|---|---|---|---|---|---|
| $y$ | Occlusion | LDA | PCA | Maha. | Group Sparse | Sparse |
| [32] | 0% | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** |
| [32] | 5% | 91.21% | 66.97% | 90.91% | 91.21% | **91.52%** |
| [32] | 10% | 88.18% | 65.15% | 88.18% | 89.70% | **90.61%** |
| [32] | 20% | 89.09% | 64.24% | 89.39% | **91.52%** | 90.91% |
| [32] | 30% | 86.06% | 64.24% | 88.18% | **90.00%** | 89.39% |
| [32] | 40% | 86.36% | 62.12% | 86.67% | 87.27% | **87.88%** |
| [32] | 50% | 85.15% | 61.52% | 83.33% | **85.76%** | 85.45% |
| [32] | 60% | 83.33% | 57.88% | 82.42% | **85.45%** | 84.55% |
| [32] | 70% | 75.76% | 59.39% | 75.15% | 79.39% | **80.61%** |
| [32] | 80% | 70.30% | 60.61% | 70.61% | **76.06%** | 73.03% |
| [32] | 90% | 47.88% | 46.36% | 47.58% | **56.67%** | 53.94% |

**Table 3.5**: Average accuracies of the method in [4] and our method for various levels of occlusion. Bold values represent the best accuracy for each row.

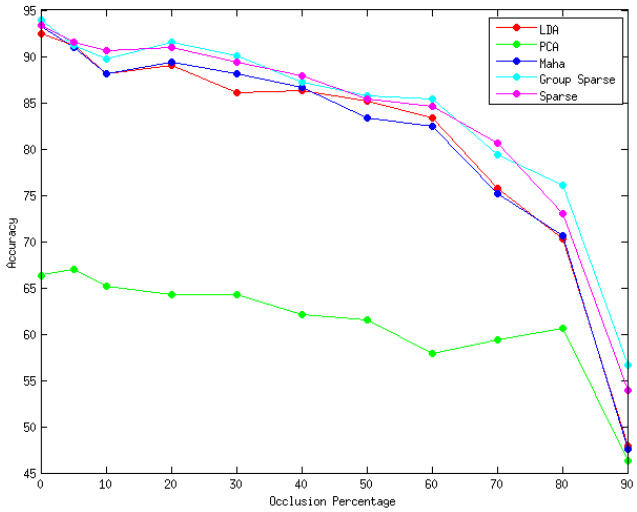


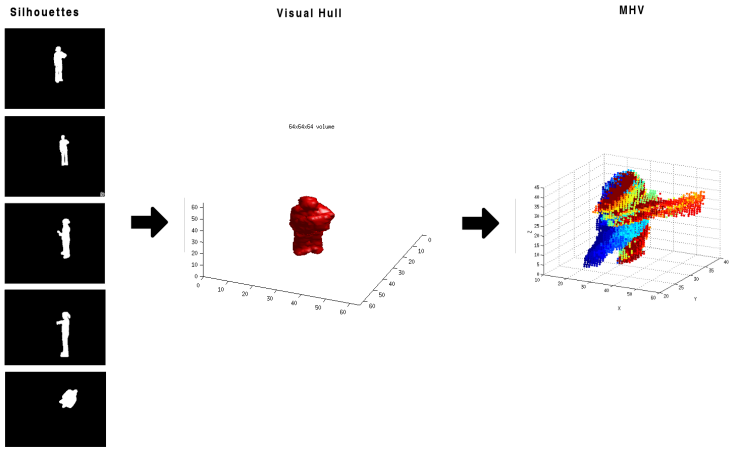

**Figure 3.4**: The plot of accuracies of the method in [4] and our method for various levels of occlusion.

## 3.2 Action Recognition in VSNs

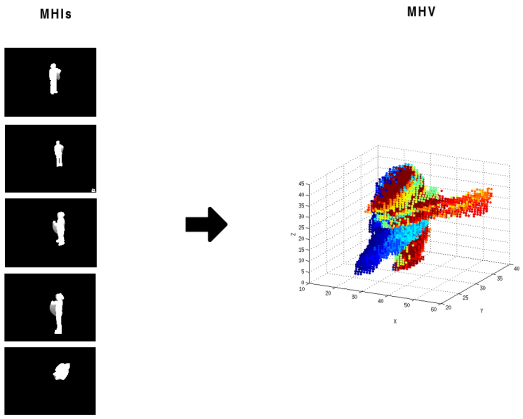
### 3.2.1 Constructing MHVs from MHIs

In [4], MHVs are constructed by concatenating visual hulls in time and visual hulls are constructed by using silhouettes obtained from multiple cameras at each time instance (see Figure 3.5-a). While constructing MHVs, only the visual hulls that involve motion are used. Therefore, at each time instance, visual hulls are created and the instances involving motion are detected. In a centralized approach, this procedure requires sending all silhouettes obtained at each time instance to a central node. Naturally, the network communication load increases by sending all silhouettes to the central node without knowing whether there is a motion at that time instance or not. Therefore, this procedure adds an unnecessary load to the communication in the network. Specifically, the load can be calculated as $Height\,of\,image \times Width\,of\,image \times Pixel\,Depth\,Bit \times Fps$. For the dataset used in Section 3.1.2, the resulting bandwidth is $291 \times 390 \times 1 \times 23 = 2.610 Mbit/s$. Considering the resource constraints of VSNs (Section 2.2.2), the way of constructing MHVs is not suitable for a setup in VSN.

In the method [4], each camera node sends the silhouettes (features) it obtained, instead of sending the raw image data. Hence, the method is suitable for decentralized action recognition (illustrated in Figure 2.8). But, as explained above, transmitting silhouettes to the fusion node will again overload the network. One may think of a distributed classification approach that can overcome this problem. But, both our sparsity-driven approach and the method in [4] is not suitable for a distributed approach in which each camera performs action recognition by the results of a neighbor




(a)




(b)

**Figure 3.5**: Flow diagram of constructing MHVs in (a) by first combining silhouettes and then visual hulls (b) by directly combining MHIs.

node and its own observations and shares the results with another neighboring node (Section 2.2.3).

To overcome the large bandwidth requirement, we have changed the way of constructing the MHVs. Instead of sending all information (silhouettes) to the fusion node at each time instance, we propose constructing motion history images (MHI) [105] at each camera and sending these images to the fusion node when there is motion in the scene (see Figure 3.5-b). Then, MHVs are constructed using MHIs similar to creating a visual hull from multiple silhouettes: i) a 3D point in the volume is set to a value greater than zero if the projection of the point to all MHIs corresponds to a value greater than zero, ii) the value of that 3D point is calculated by summing the values of its projections in MHIs. To detect the intances of motion, similar to the procedure applied on visual hulls in [4], we search for the local minima of the motion energy of MHIs. With this new line of work, there will only be a load in the network communication when a motion is detected. Since MHIs are constructed at each camera individually, one question can be about the synchronization of the detected motion instances. We think that a simple voting procedure among cameras can deal with the synchronization problem.

The duration of a motion depends on the type of the motion (walking, kicking, etc.). For this reason, the number of MHIs that will be sent to the fusion node, in other words the bandwidth required for our approach depends on the type of the motion. Thus, we approximate the bandwidth by using an average motion duration time. Specifically, assuming there is no image compression, the bandwidth can be calculated as $Height\,of\,image \times Width\,of\,image \times Pixel\,Depth\,Bit/Average\,Motion\,Time$. As-

suming that the duration of a single motion takes 3 seconds in average, the resulting bandwidth is $291 \times 390 \times 8/3 = 302.6 Kbit/s$. In other words, we save around 89% of the bandwidth.

### 3.2.2 Experimental Results

Decreasing the communication requirements so much may cause a trade-off between robustness and communication. We have tested the performance of our sparse representation framework and the method in [4] using the MHVs constructed from MHIs for both training and testing data. We have empirically set the regularization parameter in Eq. 3.8, $\lambda$, to 500 and 100 for the group sparse and sparse approaches, respectively. The resulting average rate of accuracies are given in Table 3.6. We have also presented the average rate of accuracies obtained using the original MHVs constructed from silhouettes. We have observed that when the MHVs constructed from MHIs are used, there occurs a small degradation in the average accuracies, but we still achieve a high level of accuracy (90.00%) and our sparse representation based method works better than the method in [4]. Considering the amount of saving in bandwidth, we think that the degradation is at a neglectful level. In conclusion we can say that, by changing the way of constructing features, we have adapted the action recognition method to VSN resource constraints, without degrading the performance significantly.

Similar to the experiments in Section 3.1.2, we have also performed tests using the MHVs constructed from MHIs under various conditions, such as when action descriptors are low resolution, when data are noisy, when there is occlusion. In Table 3.7, the performance of both the method in [4] and our sparse representation

| | | **The method in [4]** | | | **SR** | |
|---|---|---|---|---|---|---|
| MHVs | $y$ | LDA | PCA | Maha. | Group Sparse | Sparse |
| from Sil. | [32] | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** |
| from MHIs | [32] | 88.79% | 69.39% | 88.48% | **90.00%** | 89.70% |

**Table 3.6**: Average accuracies of the method in [4] and our method obtained by using MHVs that are constructed from MHIs.

| | **The method in [4]** | | | **SR** | |
|---|---|---|---|---|---|
| $y$ | LDA | PCA | Maha. | Group Sparse | Sparse |
| [32] | 88.79% | 69.39% | 88.48% | **90.00%** | 89.70% |
| [16] | 71.82% | 48.18% | 73.03% | **88.48%** | 87.88% |
| [8] | 15.45% | 9.09% | 9.70% | **71.82%** | 69.70% |

**Table 3.7**: Average accuracies of the method in [4] and our SR based method when MHVs constructed from MHIs are used and action descriptors are low-resolution. Bold values represent the best accuracy for each row.

framework using action descriptors in 16x16x16 and 8x8x8 voxel grid sizes are presented. We have also presented the average accuracies obtained using the 32x32x32 action descriptors are presented. The results show that while using the MHVs constructed from MHIs and the action descriptors have very low resolution, our method achieves reasonable level of accuracy. When the action descriptors have a resolution of 16x16x16, the performance of the method in [4] degrades much more dramatically than that of our method. Although the method in [4] still achieves a reasonable level of accuracy (73.03%), our method with group sparse approach achieves a better accuracy level (88.48%). For the 8x8x8 grid size, the method in [4] obtains an unacceptable level of accuracy. PCA and Mahalanobis procedures achieve only random assignment accuracies (9.09% 9.70%), whereas our method with group sparse

approach achieves a reasonable level of accuracy (71.82%). In both 16 and 8 voxel grid resolutions, our group sparse approach works slightly better than our sparse approach. These experiments demonstrate that, by construction MHVs from MHIs and using our sparse representation approach, we can both decrease the communication in the network and perform action recognition in the case of low resolution data.

We have also tested the robustness of our method for perturbations in MHVs caused by the miscalculation of the start and/or end times of the motion in MHIs. We have created the test samples by extracting the action descriptors from theMHVs, that are constructed from MHIs, corrupted with zero-mean Gaussian noise. Training samples are created by using the clean MHVs constructed from MHIs. In Table 3.8, we have presented the average accuracies achieved by the method in [4] and our method for various noise levels. The results obtained from the clean data (0% corruption) are also shown for comparison. It can be seen that for all noise levels our method outperforms the method in [4]. Even the noise variance is 90% of the maximum of the MHV, our method achieves a reasonable level of accuracy (73.64%). For all variances, we observe that group sparse and sparse versions of our approach achieve similar results. For the method in [4], there is a significant performance drop as the data become more noisy (most notably when the noise variance increases from 10% to 20%). When the noise variance is selected equal to or greater than 40% of the maximum of the MHV constructed from MHIs, the best accuracy obtained by the method in [4] is close to random assignment accuracy (9.09%). The results show that our method is robust to failures in motion detection and it can be used in VSNs without overloading the network.

| | | **The method in [4]** | | | **SR** | |
|---|---|---|---|---|---|---|
| $y$ | Corruption | LDA | PCA | Maha. | Group Sparse | Sparse |
| [32] | 0% | 88.79% | 69.39% | 88.48% | **90.00%** | 89.70% |
| [32] | 10% | 63.64% | 36.97% | 66.97% | 89.70% | **90.00%** |
| [32] | 20% | 16.67% | 17.27% | 18.18% | 87.58% | **88.48%** |
| [32] | 30% | 9.39% | 13.33% | 10.61% | **89.39%** | 88.48% |
| [32] | 40% | 9.09% | 11.21% | 9.09% | **87.88%** | 86.67% |
| [32] | 50% | 9.09% | 9.70% | 9.09% | **86.36%** | 84.55% |
| [32] | 60% | 9.09% | 9.09% | 9.09% | **85.15%** | 84.85% |
| [32] | 70% | 9.09% | 9.09% | 9.09% | 80.61% | **81.21%** |
| [32] | 80% | 9.09% | 9.09% | 9.09% | **74.85%** | 72.12% |
| [32] | 90% | 9.09% | 9.09% | 9.09% | **73.64%** | 72.73% |
| [32] | 100% | 9.09% | 9.09% | 9.09% | **67.88%** | 65.45% |

**Table 3.8**: Average accuracies of the method in [4] and our method on MHVs constructed from MHIs and corrupted by zero-mean Gaussian noise with variance specified in terms of percentages of the maximum value of the MHV. Bold values represent the best accuracy for each row.

When there is an occlusion in silhouettes, MHIs are also occluded, which also causes occlusion in MHVs. We have examined how the occlusion in MHVs constructed from MHIs affects the recognition accuracy. Test samples are created by extracting the action descriptors from the MHVs, starting from the center, occluded (set to zero) in various levels, from 5 percent to 90 percent. Training samples are created by using the clean MHVs. As in Section 3.1.2, we assume the presence of a perfect occlusion detector. In Table 3.9, the average accuracies obtained by the method in [4] and our method for various levels of occlusion are presented. The results obtained from the original data (0% occlusion) are also presented for comparison. The results show that our method performs better than the method in [4] for all levels of occlusion. For occlusion levels up to 60%, the method in [4] achieves very close accuracy rates to our method. But, for higher occlusion levels, our method is definitely better than the method in [4]. It can also be observed that the group sparse and sparse versions of our method achieve very similar accuracy levels.

Based on the experimental results we have obtained, we can say that by constructing MHVs from MHIs and using our sparse representation framework, we can perform action recognition in VSNs without overloading the network, even in limitations in data quality and quantity.

## 3.3 Role of the Sparsity Constraint in Classification Problems

Recently, the importance of sparsity constraint in Eq. 3.7 has been questioned for face recognition problem [18, 19]. It has been shown that rather than a sparsity

| | | **The method in [4]** | | | **SR** | |
|---|---|---|---|---|---|---|
| $y$ | Occlusion | LDA | PCA | Maha. | Group Sparse | Sparse |
| [32] | 0% | 88.79% | 69.39% | 88.48% | **90.00%** | 89.70% |
| [32] | 5% | 87.58% | 66.36% | 86.67% | 86.67% | **87.27%** |
| [32] | 10% | 88.88% | 68.18% | 87.27% | **87.88%** | **87.88%** |
| [32] | 20% | 87.58% | 64.85% | 87.88% | **89.39%** | 88.48% |
| [32] | 30% | 84.85% | 65.15% | 84.85% | **85.76%** | 84.85% |
| [32] | 40% | 83.03% | 63.94% | 83.03% | **87.27%** | 86.06% |
| [32] | 50% | 80.30% | 60.91% | 81.82% | 83.64% | **84.24%** |
| [32] | 60% | 81.52% | 63.64% | 80.61% | **82.73%** | 82.12% |
| [32] | 70% | 76.36% | 56.67% | 75.76% | **79.09%** | 76.36% |
| [32] | 80% | 69.70% | 56.67% | 68.18% | **74.55%** | 74.24% |
| [32] | 90% | 54.85% | 47.58% | 55.76% | **63.33 %** | 60.30% |

**Table 3.9**: Average accuracies of the method in [4] and our method using MHVs constructed from MHIs under various levels of occlusion. Bold values represent the best accuracy for each row.

constraint imposed by $l_1$ norm, using a standard $l_2$ regularization term also achieves similar level of accuracy. This is a very interesting observation that questions the necessity of imposing sparsity for classification problems. In this section, we have analyzed the role of sparsity constraint for two different action recognition problems.

In order to use the $l_2$ regularization term, mathematically, we change the optimization problem in Eq. 3.7 as the following regularized least squares problem:

$$\hat{x} = \arg\min_{x} ||y - \psi x||_2 + \lambda ||x||_2 \tag{3.15}$$

The solution of the above problem can be analytically derived as:

$$\hat{x} = \left(\psi^T \psi + \lambda \cdot I\right)^{-1} \psi^T y \tag{3.16}$$

After we find the solution of the regularized least squares problem using Eq. 3.16, we classify the test sample, $y$, by assigning it to the object class that minimizes the residual between $y$ and approximation $\tilde{y}$ as in Eq. 3.9.

### 3.3.1 Analysis for the 3-D Action Recognition Problem

By using the same features and dataset explained in Section 3.1.1 and Section 3.1.2, we have tested the performance of the classifier based on $l_2$ regularization for the 3-D action recognition problem.

Table 3.10 shows the action classification results of $l_2$ regularization and sparse representation using both sparsity and group sparsity constraints. As mentioned in Section 3.1.2, the regularization parameter in Eq. 3.8, $\lambda$, is set to 500 and 100 for the group

| | **The method in [4]** | | | **SR** | | $l_2$**-norm** |
|---|---|---|---|---|---|---|
| Action | LDA | PCA | Maha. | Group Sparse | Sparse | |
| Check watch | 83.33% | 46.66% | **86.66%** | 80.00% | 83.33% | 80.00% |
| Cross arms | **100.00%** | 83.33% | **100.00%** | **100.00%** | **100.00%** | **100.00%** |
| Scratch head | 93.33% | 46.66% | 93.33% | **96.66%** | **96.66%** | 90.00% |
| Sit down | 93.33% | 93.33% | 93.33% | **96.66%** | **96.66%** | **96.66%** |
| Get up | 90.00% | 83.33% | **93.33%** | 90.00% | 90.00% | **93.33%** |
| Turn around | **96.66%** | 93.33% | **96.66%** | **96.66%** | **96.66%** | **96.66%** |
| Walk | **100.00%** | **100.00%** | **100.00%** | **100.00%** | **100.00%** | **100.00%** |
| Wave hand | **90.00%** | 53.33% | 80.00% | 83.33% | 86.66% | 80.00% |
| Punch | 93.33% | 53.33% | **96.66%** | **96.66%** | **96.66%** | **96.66%** |
| Kick | 93.33% | 83.33% | **96.66%** | **96.66%** | **96.66%** | 93.33% |
| Pick up | 83.33% | 66.66% | **90.00%** | **90.00%** | **90.00%** | **90.00%** |
| Average | 92.42% | 66.36% | **93.33%** | **93.33%** | **93.93%** | 92.42% |

**Table 3.10**: Accuracies of the method in [4] and our methods based on sparse representation and $l_2$ regularization for each action. Bold values represent the best accuracy for each action.

sparse and sparse approaches, respectively. For the regularized least squares problem, we have set $\lambda$ to 1, $10^2$, $10^3$, $10^4$, $10^5$ and run the classifier. We have observed that we obtain the same classification accuracy for all $\lambda$ values. This implies that classification based on $l_2$ regularization does not require much effort for parameter selection. We have also presented the performance of the original method in [4].

It can be seen that we can also achieve high level of accuracy via classification based on $l_2$ regularization. For 6 actions (cross arms, sit down, turn around, walk, punch, pick up), the $l_2$ regularizer obtains the same level of accuracy with the sparse representation based classifier. Just for 4 actions (check watch, scratch head, wave hand, kick), it obtains worse accuracy rates. On the other hand, for the "get up" action, it achieves slightly better level of accuracy.

| | **The method in [4]** | | | **SR** | | $l_2$**-norm** |
|---|---|---|---|---|---|---|
| $y$ | LDA | PCA | Maha. | Group Sparse | Sparse | |
| [32] | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** | 92.42% |
| [16] | 78.18% | 44.55% | 77.27% | 90.00% | **90.61%** | 83.03% |
| [8] | 20.91% | 9.09% | 9.09% | **73.64%** | 67.88% | 71.21% |

**Table 3.11**: Average accuracies of the method in [4] and our methods based on sparse representation and $l_2$ regularization when action descriptors are low-resolution. Bold values represent the best accuracy for each row.

As in Section 3.1.2, we have tested the performance of the $l_2$ regularizer under various conditions. Table 3.11, Table 3.12 and Table 3.13 present the results obtained using action descriptors in 16x16x16 and 8x8x8 voxel grid sizes, using MHVs corrupted by zero-mean Gaussian noise with variance specified in terms of percentages of the maximum value of the MHV and using MHVs under various levels of occlusion, respectively. We have presented the performance of the original method in [4] and sparse representation based classifiers. We can observe that $l_2$ regularizer also achieves high level of accuracy in various conditions including low-resolution data, occlusion and noise. Except for action descriptors in 16x16x16 voxel grid size, highly corrupted and occluded cases, $l_2$ regularizer achieves very close level of accuracy with the sparse representation based classifier. Interestingly, for the extreme occluded case, it achieves slightly better level of accuracy.

### 3.3.2 Analysis for 2-D Action Recognition Problem

We have also analyzed the performance of classification based on $l_2$ regularization for a different action recognition problem. We have used the well-known 2-D action

| | | The method in [4] | | | SR | | $l_2$-norm |
|---|---|---|---|---|---|---|---|
| $y$ | Corruption | LDA | PCA | Maha. | Group Sparse | Sparse | |
| [32] | 0% | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** | 92.42% |
| [32] | 10% | 89.39% | 42.42% | 90.30% | 92.73% | **93.33%** | 91.82% |
| [32] | 20% | 65.15% | 16.06% | 63.03% | **93.03%** | 92.73% | 90.61% |
| [32] | 30% | 25.76% | 9.09% | 24.55% | **91.52%** | 90.61% | 87.88% |
| [32] | 40% | 11.52% | 9.09% | 13.33% | **92.42%** | **92.42%** | 90.00% |
| [32] | 50% | 11.21% | 9.09% | 12.42% | **90.91%** | 90.30% | 86.06% |
| [32] | 60% | 10.00% | 9.09% | 10.30% | 89.39% | **90.30%** | 86.67% |
| [32] | 70% | 10.61% | 9.09% | 10.00% | **86.67%** | 86.36% | 79.39% |
| [32] | 80% | 9.70% | 9.09% | 9.39% | **83.94%** | **83.94%** | 76.06% |
| [32] | 90% | 9.70% | 9.09% | 9.09% | **81.82%** | 80.91% | 75.45% |
| [32] | 100% | 9.39% | 9.09% | 9.09% | **84.55%** | 83.94% | 70.00% |

**Table 3.12**: Average accuracies of the method in [4] and our methods based on sparse representation and $l_2$ regularization on data corrupted by zero-mean Gaussian noise. Bold values represent the best accuracy for each row.

recognition method in [10] that is based on space-time interest points and bag-of-features model. In the following subsections, we describe the details of this method and present the result of the analysis on the role of sparsity constraint.

### Space-time features and Bag-of-features Model

The method in [10] is based on the idea that the Harris interest point detector in the spatial domain can be extended into the spatio-temporal domain by requiring the image values in space-time. Points with such properties are defined as spatial interest points that have distinct locations in time corresponding to local spatio-temporal neighborhoods with non-constant motion.

To model a spatio-temporal image sequence, a function $f : R^2 \times R \rightarrow R$ is used

| | | **The method in [4]** | | | **SR** | | $l_2$**-norm** |
|---|---|---|---|---|---|---|---|
| $y$ | Occlusion | LDA | PCA | Maha. | Group Sparse | Sparse | |
| [32] | 0% | 92.42% | 66.36% | 93.33% | 93.33% | **93.93%** | 92.42% |
| [32] | 5% | 91.21% | 66.97% | 90.91% | 91.21% | **91.52%** | 90.91% |
| [32] | 10% | 88.18% | 65.15% | 88.18% | 89.70% | **90.61%** | 89.70% |
| [32] | 20% | 89.09% | 64.24% | 89.39% | **91.52%** | 90.91% | 88.18% |
| [32] | 30% | 86.06% | 64.24% | 88.18% | **90.00%** | 89.39% | 88.48% |
| [32] | 40% | 86.36% | 62.12% | 86.67% | 87.27% | **87.88%** | 85.76% |
| [32] | 50% | 85.15% | 61.52% | 83.33% | **85.76%** | 85.45% | 84.85% |
| [32] | 60% | 83.33% | 57.88% | 82.42% | **85.45%** | 84.55% | 79.39% |
| [32] | 70% | 75.76% | 59.39% | 75.15% | 79.39% | **80.61%** | 74.85% |
| [32] | 80% | 70.30% | 60.61% | 70.61% | **76.06%** | 73.03% | 63.33% |
| [32] | 90% | 47.88% | 46.36% | 47.58% | 56.67% | 53.94% | **57.88%** |

**Table 3.13**: Average accuracies of the method in [4] and our methods based on sparse representation and $l_2$ regularization for various levels of occlusion. Bold values represent the best accuracy for each row.

and its linear scale-space representation $L : R^2 \times R \times R^2_+ \rightarrow R$ is constructed by convolution of the image sequence, $I(x, y, t)$, with an anisotropic Gaussian kernel with distinct spatial variance $\sigma_l^2$ and temporal variance $\tau_l^2$

$$L(\cdot; \sigma_l^2, \tau_l^2) = g(\cdot; \sigma_l^2, \tau_l^2) * I(\cdot), \tag{3.17}$$

where the spatio-temporal separable Gaussian kernel is defined as

$$g(x, y, t, ; \sigma_l^2, \tau_l^2) = \frac{exp(-(x^2 + y^2)/2\sigma_l^2 - t^2/2\tau_l^2)}{\sqrt{(2\pi)^3 \sigma_l^4 \tau_l^2}} \tag{3.18}$$

In [10], similar to the Harris point detector, the spatio-temporal second-moment matrix, which is a 3-by-3 matrix composed of first order spatial and temporal derivatives

averaged with a Gaussian weighting function $g(\cdot;\sigma_i^2,\tau_i^2)$, is considered.

$$\mu = g(\cdot;\sigma_i^2,\tau_j^2) * \begin{pmatrix} L_x^2 & L_xL_y & L_xL_t \\ L_xL_y & L_y^2 & L_yL_t \\ L_xL_t & L_yL_t & L_t^2 \end{pmatrix} \tag{3.19}$$

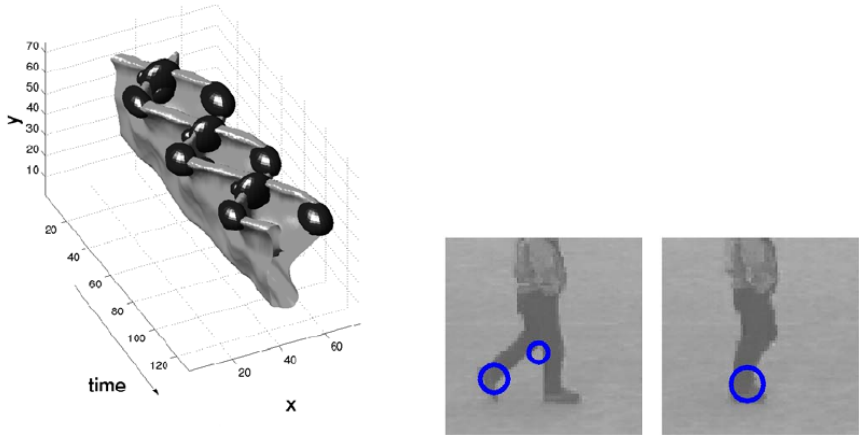
where the integration scales are in multiple-levels, $\sigma_i^2 = 2^{(1+i)/2}, i = 1, \cdots, 6$ and $\tau_j^2 = 2^{j/2}, j = 1, 2$, while the first-order derivatives are defined as $L_\xi(\cdot;\sigma_l^2,\tau_l^2) = \partial_\xi(g * I)$.

Interest points are detected by searching for regions in $I$ having significant eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of $\mu$. In order to find such regions, the Harris corner function defined for the spatial domain is extended into the spatio-temporal domain by combining the determinant and the trace of $\mu$ in the following way

$$H = det(\mu) - k\text{trace}^3(\mu) = \lambda_1\lambda_2\lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3 \tag{3.20}$$

In [10], they define the ratios $\alpha = \lambda_2/\lambda_1$ and $\beta = \lambda_3/\lambda_1$ (assuming $\lambda_1 \leq \lambda_2 \leq \lambda_3$). To show that the positive local maxima of $H$ correspond to points with high values of $\lambda_1, \lambda_2, \lambda_3$, they rewrite $H = \lambda_1^3(\alpha\beta k(1 + \alpha + \beta)^3)$. Then, the requirement $H \geq 0$ implies that $k \leq \alpha\beta/(1 + \alpha + \beta)^3$ and it follows that as $k$ increases towards its maximal value $k = 1/27$, both ratios $\alpha$ and $\beta$ tend to one. For sufficiently large values of $k$, positive local maxima of $H$ correspond to points with high variation of the image gray-values in both the spatial and the temporal dimensions. Thus, spatio-temporal interest points of $I$ can be found by detecting local positive spatio-temporal maxima in $H$. An example of interest points detected by this approach for walking motion are presented in Figure 3.6.

To characterize motion and appearance of local features, they compute histogram descriptors of space-time volumes in the neighborhood of detected points in [10]. The




(a) (b)

**Figure 3.6**: Space-time features detected for a walking pattern: (a) 3-D plot of a spatio-temporal leg motion (up side down) and corresponding features (in black); (b) Features overlaid on selected frames of a sequence.

size of each volume ($\delta_x$ , $\delta_y$ , $\delta_t$ ) is related to the detection scales by $\delta_x, \delta_y = 2m\sigma$, $\delta_t = 2m\tau$ (they set $m$ to 9 in the experiments). For each volume they compute coarse histograms of oriented gradient (HoG) and optic flow (HoF). Then, normalized histograms are concatenated into HoG and HoF descriptor vectors.

Given a set of spatio-temporal features, extracted as above, they build a spatio-temporal bag-of-features (BoF) model. This requires the construction of a visual vocabulary. In the experiments they cluster a subset of $100k$ features sampled from the training videos with the k-means algorithm. They empirically set the number of clusters to $k = 4000$. The BoF representation then assigns each feature to the closest (using the Euclidean distance) vocabulary word and computes the histogram of visual word occurrences over a space-time volume corresponding to the entire video sequence. This histogram is used as a descriptor to represent the on going motion in

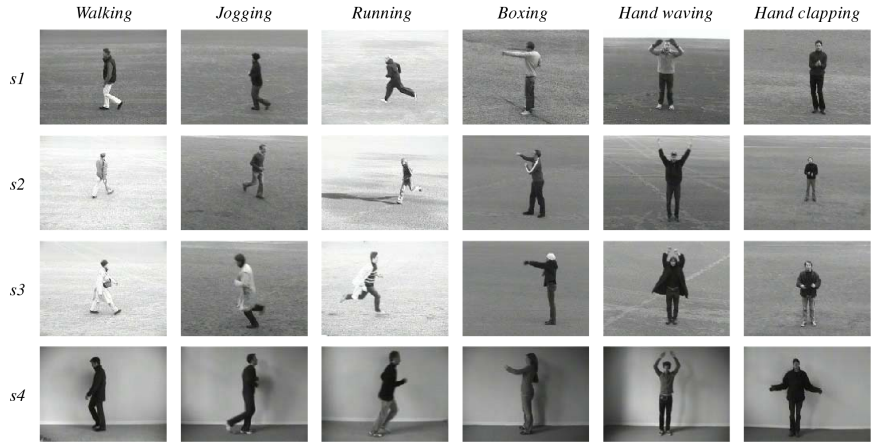



**Figure 3.7**: Examples of sequences corresponding to different types of actions and scenarios in KTH-dataset [5].

the video sequence.

### Results

In the experiments, we have used the KTH-dataset in [5] containing six types of human actions (walking, jogging, running, boxing, hand waving and hand clapping) performed several times by 25 subjects in four different scenarios: outdoors s1, outdoors with scale variation s2, outdoors with different clothes s3 and indoors s4. Sample images from the dataset is presented in Figure 3.7. All sequences are taken over homogeneous backgrounds with a static camera with 25fps frame rate. The sequences are downsampled to the resolution of 160 $\times$ 120 pixels and have a length of four seconds in average.

In [10, 5], the spatio-temporal BOF descriptors are classified using nearest-neighbor (NN) classifier and a non-linear support vector machine (SVM) with a multi-channel

$\chi^2$ kernel. For NN, we have used 16 subjects for training and 9 subjects for testing. For SVM, we divided the dataset into a training set (8 people), a validation set (8 people) and a test set (9 people). SVM is trained on a training set while the validation set is used to optimize its parameters. The presented recognition results are obtained on the test set. Similarly, we have used 16 people for training and 9 people for testing in our sparse representation based classifier with both group sparse and sparse approach and in the $l_2$ regularizer based classifier. The regularization parameter in Eq. 3.8 and Eq.3.16, $\lambda$, is empirically set to 0.1 for both group sparse, sparse and $l_2$ approaches. For $l_2$ regularizer based classifier, again we have observed that we obtain the same classification accuracy for different $\lambda$ values.

Table 3.14 shows the action recognition results for all methods. It can be seen that both three approaches (group sparse, sparse and $l_2$) achieve better level of accuracies on average compared to the method in [10]. Among these three approaches, the sparse approach works slightly better than the others. We can also see that $l_2$ regularization based classifier again achieves high level of accuracy. Except the boxing action, it achieves identical results with the sparse approach. Interestingly, it obtains the identical level of accuracies with the group sparse approach. Only for hand clapping action, SVM works slightly better than all other methods.

### 3.3.3 Conclusion on Analysis for the Role of Sparsity

In Section 3.3.1 and 3.3.2, we have analyzed the role of sparsity constraint for two different action recognition problems. In the light of these experimental results, we

| | **The method in [10]** | | **SR** | | $l_2$**-norm** |
|---|---|---|---|---|---|
| Action | NN | SVM | Group Sparse | Sparse | |
| Boxing | **97.22%** | 97.00% | 94.44% | **97.22%** | 94.44% |
| Hand Clapping | 94.29% | **95.00%** | 94.29% | 94.29% | 94.29% |
| Hand Waving | **97.22%** | 91.00% | **97.22%** | **97.22%** | **97.22%** |
| Jogging | 91.67% | 89.00% | **94.44%** | **94.44%** | **94.44%** |
| Running | 66.67% | 80.00% | **83.33%** | **83.33%** | **83.33%** |
| Walking | 97.22% | 99.00% | **100.00%** | **100.00%** | **100.00%** |
| Average | 90.71% | 91.83% | 93.96% | 94.42% | 93.96% |

**Table 3.14**: Accuracies of the method in [10] and our methods based on sparse representation and $l_2$ regularization for each action. Bold values represent the best accuracy for each action.

can say that sparse representation based classification is an important technique that outperforms existing action recognition methods especially in the case of limitations in data quality and quantity. On the other hand, although we could not see a case in which it works better than sparse representation based classifier, we have seen that $l_2$ regularization based classification also achieves high level of accuracies.

There is a common idea behind both approaches: to express a test sample as a linear combination of training samples. Experimental results also show that this common idea is the trigger force that provides robust action recognition performance. In order to have a complete conclusion, we believe that this analysis should be performed for different classification problems other than action recognition problem which are not in the scope of this thesis.

# 4 HUMAN TRACKING IN VSNS VIA FEATURE COMPRESSION

In this chapter, we describe our feature compression framework proposed to overcome communication problems of human tracking systems in visual sensor networks. This framework is another contribution of this thesis. As we have seen in Section 2.2.3, because of the energy and bandwidth constraints in VSNs, rather than centralized approaches, decentralized and distributed approaches have been proposed [76]. Following these studies, we propose a decentralized approach in which a feature compression framework is used to reduce the communication in the network. In Section 4.1, we describe our decentralized approach and the multi-camera tracking algorithm that we use in our framework.

Section 4.2 presents our feature compression framework. Instead of directly sending features to the fusion node, block-based compression is performed on features by transforming each block to an appropriate domain. Then, only the significant coefficients in this new representation are sent to the fusion node. Here, we perform goal-directed compression. In the tracking context, this is achieved by performing local processing at the nodes and compressing the resulting features which are related to the tracking goal, rather than compressing raw images. To the best of our

knowledge, compression of features computed in the context of tracking in a VSN has not been proposed in previous work.

The selection of the domain that we transform the blocks of features is very important. The properties of an appropriate domain is discussed in Section 4.3. In order to choose an appropriate domain, we have performed a comparison between well-known transforms. In Section 4.4, both qualitative and quantitative results together with our analysis on domain selection are provided.

## 4.1 Overview

### 4.1.1 Decentralized Human Tracking

As we have described in Section 2.2.3, in a traditional setup of camera networks (centralized tracking), each camera acquires an image and sends this raw data to a central unit. In the central unit, relevant features are extracted from multi-view data, by using these features, the positions of the humans are estimated. Hence, integration of multi-view information is done in raw-data level by pooling all images in a central unit. With a data compression perspective, the common approach to get over high-bandwidth requirements is to compress images and collect them in a central unit to perform the tasks of interest. In this strategy, the main goal is to focus on low-level communication. The communication load is decreased by compressing the raw data without regard to the final inference goal based on the information content of the data. Since such a strategy will affect the quality of the transmitted data, it may decrease the performance of further inference tasks.

The decentralized approaches fit very well to VSNs in many aspects. The processing capability of each camera is utilized by performing feature extraction at camera-level. Since cameras are grouped into clusters, the communication overhead is reduced by limiting the cooperation within each cluster and among fusion nodes. In other words, by a decentralized approach, feature extraction and communication are distributed among cameras in clusters, therefore, efficient estimation can be performed. One may argue that distributed approaches have more ability to decrease communication than decentralized methods. Although, in theory this is true, in practice distributed methods require an inference approach totally different from decentralized methods. Therefore, we need to propose tracking algorithms based on distributed estimation. On the other hand, by using decentralized strategy, it is easier to have decentralized trackers by making small modifications on centralized trackers. For this reason, we base our framework on decentralized tracking.

Modeling the dynamics of humans in a probabilistic framework is a common perspective of many multi-camera human tracking methods [6, 110, 111, 112]. In tracking methods based on a probabilistic framework, data and/or extracted features are represented by likelihood functions, $p(y|x)$ where $y \in R^d$ and $x \in R^m$ are the observation and state vectors, respectively. Human tracking is performed by estimating the posterior probability of state of humans given observations:

$$p(x|y) \propto p(y|x)p(x) \tag{4.1}$$

When we have $C$ cameras, assuming the observations from cameras ($y = \{y_1, \cdots, y_C\}$) are independent, we obtain the likelihood function by taking the product of marginal

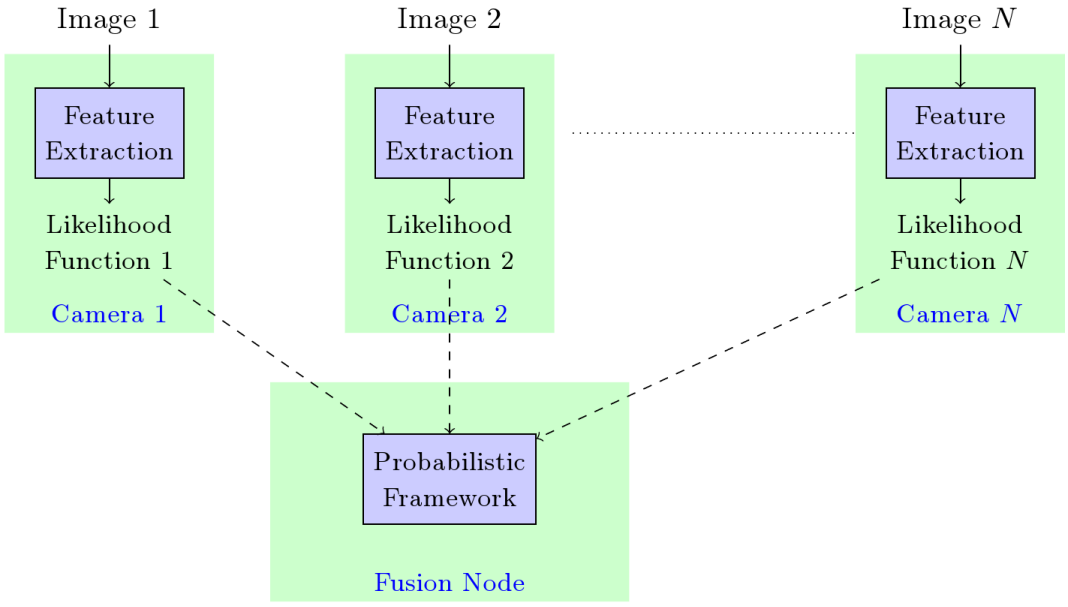



**Figure 4.1**: The flow diagram of our decentralized tracker.

distributions

$$p(y|x) = \prod_{c=1}^{C} p(y_c|x) \tag{4.2}$$

In other words, for each camera, a likelihood function is defined in terms of the observations obtained from its field of view. In centralized tracking, of course, the likelihood functions are computed after collecting the image data of each camera at the central unit. For a decentralized approach, since each camera node extracts local features from its field of view, these likelihood functions, $p(y_c|x)$, can be evaluated at the camera nodes and they can be sent to the fusion node. Then, in the fusion node the likelihoods can be combined (Eq. 4.2) and tracking can be performed in the probabilistic framework (Eq. 4.1). A flow diagram of the decentralized approach is illustrated in Figure 4.1.

### 4.1.2 The Tracking Algorithm

In this section we describe the tracking method of [6], as we apply our decentralized approach within in the context of this method. In [6], the visible part of the ground plane is discretized into a finite number $G$ of regularly spaced 2D locations. Let $\mathbf{L}_t = (L_t^1, ..., L_t^{N^*})$ be the locations of individuals at time $t$, where $N^*$ stands for the maximum allowable number of individuals. Given $T$ temporal frames from $C$ cameras, $\mathbf{I} = (\mathbf{I}_1, ..., \mathbf{I}_T)$ where $\mathbf{I}_t = (I_t^1, ..., I_t^C)$, the goal is to maximize the posterior conditional probability:

$$P(\mathbf{L}^1 = \mathbf{l}^1, ..., \mathbf{L}^{N^*} = \mathbf{l}^{N^*}|\mathbf{I}) = P(\mathbf{L}^1 = \mathbf{l}^1|\mathbf{I})$$
$$\prod_{n=2}^{N^*} P(\mathbf{L}^n = \mathbf{l}^n|\mathbf{I}, \mathbf{L}^1 = \mathbf{l}^1, ..., \mathbf{L}^{n-1} = \mathbf{l}^{n-1}) \tag{4.3}$$

where $\mathbf{L}^n = (L_1^n, ..., L_T^n)$ is the trajectory of person $n$. Simultaneous optimization of all the $L^i$s would be intractable. Instead, one trajectory after the other is optimized. $\mathbf{L}^n$ is estimated by seeking the maximum of the probability of both the observations and the trajectory ending up at location $k$ at time $t$:

$$\Phi_t(k) = \max_{l_1^n, ..., l_{t-1}^n} P(\mathbf{I}_1, L_1^n = l_1^n, ..., \mathbf{I}_t, L_t^n = k) \tag{4.4}$$

Under a hidden Markov model, the above expression turns into the classical recursive expression:

$$\Phi_t(k) = \underbrace{P(\mathbf{I}_t|L_t^n = k)}_{Appearance\ model} \max_{\tau} \underbrace{P(L_t^n = k|L_{t-1}^n = \tau)}_{Motion\ model} \Phi_{t-1}(\tau) \tag{4.5}$$

The motion model $P(L_t^n = k|L_{t-1}^n = \tau)$ is a distribution into a disc of limited radius and center $\tau$, which corresponds to a loose bound on the maximum speed of a walking

human.

From the input images $\mathbf{I}_t$, by using background subtraction, foreground binary masks, $\mathbf{B}_t$, are obtained. Let the colors of the pixels inside the blobs are denoted as $\mathbf{T}_t$ and $X_k^t$ be a Boolean random variable denoting the presence of an individual at location $k$ of the grid at time $t$. It is shown in [6] that the appearance model in Eq. 4.5 can be decomposed as:

$$\overbrace{P(\mathbf{I}_t|L_t^n=k)}^{Appearance\ model} \propto \underbrace{P(L_t^n=k|X_k^t=1,\mathbf{T}_t)}_{Color\ model} \underbrace{P(X_k^t=1|\mathbf{B}_t)}_{Ground\ plane\ occupancy} \tag{4.6}$$

In [6], humans are represented as simple rectangles and these rectangles are used to create synthetic ideal images that would be observed if people were at given locations. Within this model, the ground plane occupancy is approximated by measuring the similarity between ideal images and foreground binary masks.

Let $T_t^c(k)$ denote the color of the pixels taken at the intersection of the foreground binary mask, $B_t^c$, from camera $c$ at time $t$ and the rectangle $A_k^c$ corresponding to location $k$ in that same field of view. Say we have the reference color distributions (histograms) of the $N^*$ individuals present in the scene, $\mu_1^c, ..., \mu_{N^*}^c$. The color model of person $n$ in Eq. 4.6 can be expressed as:

$$\begin{aligned}\overbrace{P(L_t^n=k|X_k^t=1,\mathbf{T}_t)}^{Color\ model} \propto P(\mathbf{T}_t|L_t^n=k) &= P(T_t^1(k),...,T_t^C(k)|L_t^n=k) \\ &= \prod_{c=1}^{C} P(T_t^c(k)|L_t^n=k)\end{aligned} \tag{4.7}$$

In [6], by assuming the pixels whose colors are represented by $T_t^c(k)$ are independent, $P(T_t^c(k)|L_t^n = k)$ is evaluated by a product of the marginal color distribution $\mu_n^c$ at each pixel,– $P(T_t^c(k)|L_t^n = k) = \prod_{r \in T_t^c(k)} \mu_n^c(r)$. In this approach, a patch with constant color intensity corresponding to the the mode of the color distribution would be most likely. Hence, this approach may fail to capture the statistical color variability represented by the full probability density function estimated from a spatial patch. Instead, we represent $P(T_t^c(k)|L_t^n = k)$ by comparing the observed and reference color distributions, which is a well known approach used in many computer vision methods [113, 114, 115]. In particular, we compare the estimated color distribution (histogram) of the pixels in $T_t^c(k)$ and the color distribution $\mu_n^c$ with a distance metric – $P(T_t^c(k)|L_t^n = k) \propto exp(-S(H_t^{c,k}, \mu_n^c))$ where $H_t^{c,k}$ denotes the histogram of the pixels in $T_t^c(k)$ and $S(.)$ is a distance metric. As a distance metric, we use the Bhattacharya coefficient between two distributions. In this way, we can evaluate the degree of match between the intensity distribution of an observed patch and the reference color distribution.

By performing a global search with dynamic programming using Eq. 4.5, the trajectory of each person can be estimated.

## 4.2 Compressing Likelihood Functions

The bandwidth required for sending local likelihood functions depends on the size of likelihoods (i.e., the number of "pixels" in a 2D likelihood function) and the number of cameras in the network. To make the communication in the network feasible, we propose a feature compression framework. In our framework, similar to image

compression, we compress the likelihood functions by transforming them to a proper domain and keeping only the significant coefficients, assuming significant parts of the likelihood functions are sufficient for performing tracking. At each camera node, we first split the likelihood function into blocks. Then, we transform each block to a proper domain and take only the significant coefficients in the new representation. Instead of sending the function itself, we send this new representation of each block. In this way, we reduce the communication in the network.

Mathematically, we have the following linear system:

$$y_c^b = A \cdot x_c^b \tag{4.8}$$

where $y_c^b$ and $x_c^b$ represent the $b$th block of the likelihood function of camera $c$ (for a person of interest in a particular time instant, $P(T_t^c(k)|L_t^n = k)$ in Eq. 4.7) and its representation, respectively, and $A$ is the domain we transform $y_c^b$ to. In most of the compression methods, the matrix $A$ is chosen to be a unitary matrix. Hence, we can obtain $x_c^b$ by multiplying $y_c^b$ with the Hermitian transpose of $A$:

$$x_c^b = A^* \cdot y_c^b \tag{4.9}$$

Figure 4.2 illustrates our likelihood compression scheme. Based on existing work on VSN hardware platforms [116, 117, 118, 119], it is reasonable to expect that the computational power of cameras in VSNs is sufficient for performing the camera-level computations required by our approach.

Notice that in our feature compression framework, we do not require the use of specific image features or likelihood functions. The only requirement is that the tracking

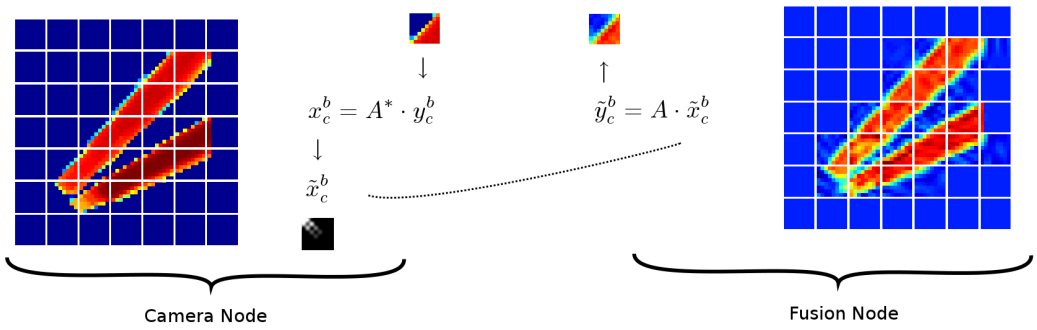



**Figure 4.2**: Our Likelihood compression scheme. On the left, there is a local likelihood function ($P(T_t^c(k)|L_t^n = k)$ in Eq. 4.7). First, we split the likelihood into blocks, then we transform each block to the domain represented by matrix $A$ and obtain the representation $x_c^b$. We only take significant coefficients in this representation and obtain a new representation $\tilde{x}_c^b$. For each block, we send this new representation to fusion node. Finally, by reconstructing each block we obtain the whole likelihood function on the right.

method should be based on a probabilistic framework, which is a common approach for modeling the dynamics of humans. Hence, our framework is a generic framework that can be used with many probabilistic tracking algorithms in a VSN environment.

In all camera nodes and fusion nodes, the matrix $A$ is common, therefore, at the fusion node, likelihood functions of each camera can be reconstructed simply by multiplying the new representation with the matrix $A$. In general, this may require an offline coordination step to decide the domain that is matched with the task of interest. In the next subsection, we go through the question of which domain should be selected in Eq. (4.8).

## 4.3 A Proper Domain for Compression

By sending the compressed likelihoods to the fusion node, our goal is to decrease the communication in the network without affecting the tracking performance significantly. On one hand, we want to send less coefficients, on the other hand, we do not want to decrease the quality of the likelihoods, i.e., we want to have small reconstruction error. For this reason, we need to select a domain that is well-matched to the likelihood functions, providing the opportunity to accurately reconstruct the likelihoods back using a small number of coefficients.

Image compression using transforms is a mature research area. Numerous transforms such as the discrete cosine transform (DCT), the Haar transform, symmlets, coiflets have been proposed and proven to be successful [120, 121, 122]. DCT is a well-known transform that has the ability to analyze non-periodic signals. Haar wavelet is the first known wavelet basis that consists of orthonormal functions. In wavelet theory, *number of vanishing moments* and *size of support* are two important properties that affect the ability of wavelet bases to approximate a particular class of functions with few non-zero wavelet coefficients [123]. In order to reconstruct likelihoods accurately using from a small number of coefficients, we wish wavelet functions to have large number of vanishing moments and small size of support. Coiflets [124] are a wavelet basis with large number of vanishing moments and Symmlets [125] are a wavelet basis that have minimum size of support. The performance of these domains has been analyzed in the context of our experiments and a proper domain has been selected accordingly as described in Section 4.4.2.

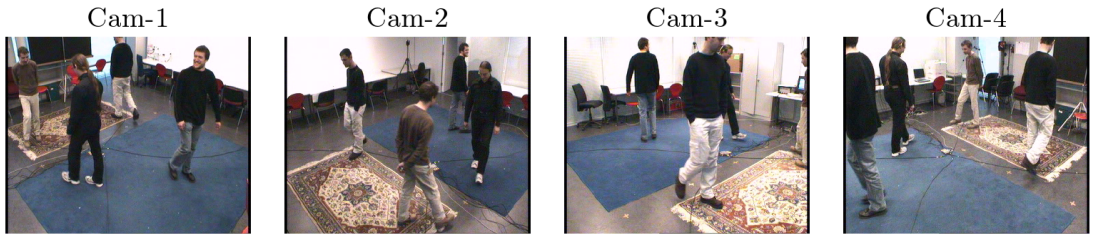
## 4.4 Experimental Results

### 4.4.1 Setup

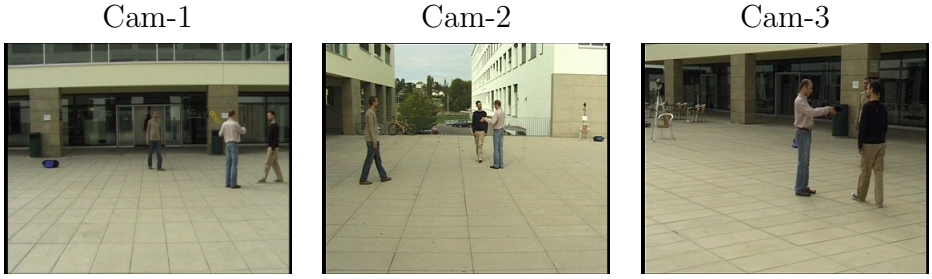
In the experiments, we have simulated the VSN environment by using the indoor and outdoor multi-camera dataset in [6]. Indoor dataset includes four people sequentially entering a room and walking around. The sequence was shot by four synchronized cameras in a 50 $m^2$ room. The cameras were located at each corner of the room. In this sequence, the area of interest was of size 5.5 $m\times$ 5.5 $m \simeq 30\ m^2$ and discretized into $G = 56 \times 56 = 3136$ locations, corresponding to a regular grid with a $10cm$ resolution. Outdoor dataset was shot in university campus and it includes up to four individuals appear simultaneously. This sequence was shot by three synchronized cameras. The area of interest at this sequence is of size 10 $m\times$ 10 $m$ and discretized into $G = 40 \times 40 = 1600$ locations, corresponding to a regular grid with a resolution of $25cm$.

For the correspondence between camera views and the top view, the homography matrices provided with the dataset are used. The size of the images are $360 \times 288$ pixels and the frame rate for all of the cameras is 25 fps. A sample set of images for indoor and outdoor dataset is shown in Figure 4.3.

From the formulation in Section 4.1.2, we can see that there are two different likelihood functions defined in the tracking method [6]. One is the ground plane occupancy map (GOM), $P(X_k^t = 1|\mathbf{B}_t)$, approximated using the foreground binary masks. The other is the ground plane color map (GCM), $P(L_t^n = k|X_k^t = 1, \mathbf{T}_t)$, which is a multi-view color likelihood function defined for each person individually. This map is




(a)




(b)

**Figure 4.3**: A sample set of images from (a) indoor and (b) outdoor multi-camera datasets [6].

obtained by combining the individual color maps, $P(T_t^c(k)|L_t^n = k)$, evaluated using the images each camera acquired. Since foreground binary masks are simple binary images that can be easily compressed by a lossless compression method, they can be directly sent to the fusion node without overloading the network. Therefore, we keep these binary images as in the original method and GOM is evaluated at the fusion node. In our framework, we evaluate GCM in a decentralized way (as presented in Figure 4.1): At each camera node ($c = 1, \cdots, C$), the local color likelihood function for the person of interest ($P(T_t^c(k)|L_t^n = k)$) is evaluated by using the image acquired from that camera. Then, these likelihood functions are sent to the fusion node. At the fusion node, these likelihood functions are integrated to obtain the multi-view color likelihood function (GCM) (Eq. 4.7). By combining GCM and GOM with the motion model, the trajectory of the person of interest is estimated at the fusion node using dynamic programming (Eq. 4.5). The whole process is run for each person in the scene.

Fusion node selection and sensor resource management (sensor tasking) is out of scope of this thesis. We have assumed that one of the camera nodes, relatively more powerful one, has been selected as the fusion node. Since each camera keeps a reference color histogram individually for each person in the scene, data association between different people is performed at the camera level. Then, at the fusion node, assuming there is only one person in the scene in the beginning of the tracking process, we assign an ID number for each likelihood function coming from cameras to the fusion node. Likelihoods with the same ID number from different cameras are associated with one another at the fusion node.

### 4.4.2 Comparison of Domains

As discussed in Section 4.3, it is very important to select a domain (matrix $A$ in Eq. (4.8)) that can compress the likelihood functions effectively. To select a proper domain, we have performed a comparison between DCT, Haar, Symmlet, and Coiflet domains and examined the errors in reconstructing the likelihoods using various number of coefficients. For the Symmlet domain, the size of support is set to 8 and for the Coiflet domain, the number of vanishing moments is set to 10. In the comparison, we have used 20 different likelihood functions obtained from the tracker in [6]. We have also analyzed the effect of block size by choosing two different block sizes: 8×8 and 4×4. After we transform each block to a domain, we have reconstructed the blocks by using only 1, 2, 3, 4, 5, and 10 most significant coefficient(s). In total, for a block size of 8×8, taking the most significant 2 coefficients results in 98 coefficients overall. According to the structure of the likelihood functions, the elements in a block may all be zero. For such a block all the coefficients will be zero, thereby we do not need to take coefficients. Thus, we may end up with even smaller number of coefficients.

Figure 4.4 shows the average of reconstruction errors of each domain for different block sizes. As explained above, the total number of significant coefficients used for reconstruction may change depending on the structure of likelihoods. For this reason, the x-axis in Figure 4.4 corresponds to the average of number of coefficients obtained by taking the 1, 2, 3, 4, 5 and 10 most significant coefficient(s) per block. We can see that using DCT with a block size of 8×8 outperforms other domains. Following this observation, in our tracking experiments, this setting has been used.

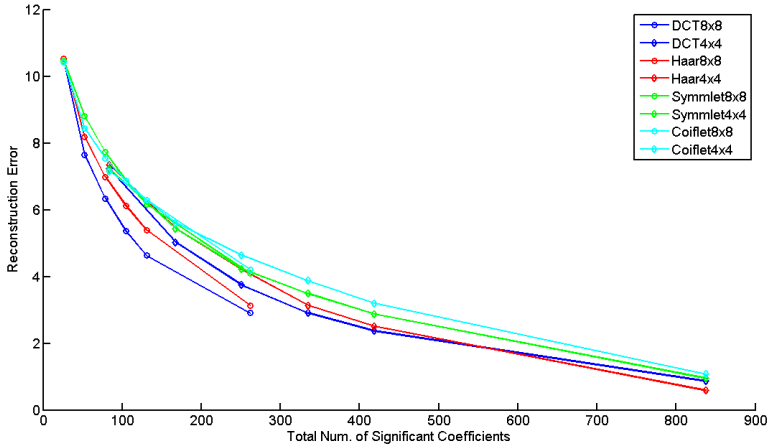



**Figure 4.4**: The average reconstruction errors of DCT, Haar, Symmlet, and Coiflet domain for block sizes of 8×8 and 4×4 using 1, 2, 3, 4, 5 and 10 most significant coefficient(s) per block.

### 4.4.3 Indoor Tracking Results

In this subsection, we present the performance of our method used for multi-view multi-person tracking. In the experiments, we have compared our method with the traditional centralized approach of compressing raw images and a decentralized method in which, similar to [7], a Kalman filter is used in the fusion node to estimate the position of a person in the scene using the observations coming from cameras. In the centralized approach, after the raw images are acquired by the cameras, similar to JPEG compression, each color channel in the images are compressed and sent to the central node. In the central node, features are extracted from the reconstructed images and tracking is performed using the method in [6]. In the decentralized method, after likelihood functions are computed, each camera sends the peak point of the distribution to the fusion node as observation. In the fusion node, the observations

of each camera are spatially averaged and using the average position as observation, a Kalman filter is applied to estimate the position of the person on ground plane. The position of all people in the scene is estimated by running an individual Kalman filter for each person.

For both our method and the centralized approach we have used DCT domain with a block size of 8×8 and took only the 1, 2, 3, 4, 5, 10, and 25 most significant coefficient(s). Consequently, in our method with the likelihoods of 56×56 size, at each camera in total we end up with at most 49, 98, 147, 196, 245, 490 and 1225 coefficients per person. Since there are four individuals in the scene at maximum, each camera sends at most 196, 392, 588, 784, 980, 1960 and 4900 coefficients. As mentioned in the previous section, these are the maximum number of coefficients, since there may be some all-zero blocks. To make a fair comparison, in the centralized approach we compress the images with 360×288 size and 3 color channels. Hence, at each camera we end up with 4860, 9720, 14580, 19440, 24300, 48600 and 121500 coefficients. In the decentralized method that uses Kalman filter, for each person, each camera sends only 2 points, xy-point of the peak point, to the fusion node. In total, we end up with 8 points in maximum for four individuals.

A groundtruth for this sequence is obtained by manually marking the people on ground plane, in intervals of 25 frames. Tracking errors are evaluated via Euclidean distance between the tracking and manual marking results (in intervals of 25 frames). Figure 4.5 presents the average of tracking errors over all people versus the total number of significant coefficients used in communication for the centralized approach and for our method. Since the total number of significant coefficients sent by a camera

in our method may change depending on the structure of likelihood functions and the number of people at that moment, the maximum is shown in Figure 4.5. It can be clearly seen that the centralized approach is not capable of decreasing the communication without affecting the tracking performance. It needs at least 121500 significant coefficients in total to achieve an error of around 1 pixel in the grid on average. On the other hand, our method, down to using 3 significant coefficients per block, achieves an error of around 1 pixel in the grid on average. In our experiments, this led to sending at most 408 coefficients for four people. Taking less than 3 coefficients per block affects the performance of the tracker and produces an error of 11.5 pixels in the grid on average. But in overall, our method significantly outperforms the centralized approach. In Figure 4.5, we also present the performance of the decentralized Kalman approach. We can see that, by using this approach, we can obtain a huge reduction in the communication, but we cannot perform robust tracking. Our framework is also advantageous over an ordinary decentralized approach that directly sends likelihood functions to the fusion node. In such an approach, we send each data point in the likelihood function, resulting a need of sending 12544 values for tracking four people. The performance of this approach is also given in Figure 4.5. It can be seen that we can both achieve the same level of tracking accuracy and decrease the communication in the network.

The tracking errors for each person and the tracking results, obtained by the centralized approach using 48600 coefficients in total, are given in Figure 4.6-a and Figure 4.6-b, respectively. It can be seen that although the centralized approach can track the first and the second individuals very well, there is an identity association problem for the third and fourth individuals. Figure 4.7 shows the tracking errors of each

person and tracking results obtained by the decentralized Kalman approach. It can be seen that it fails to track the people in the scene. Nearly for all people, there occurs identity association problems. In some frames, it loses the track of the person and starts tracking a virtual person in the scene (frame no. 1173 in Figure 4.7-b). The reason of these failures are that the amount of information coming from cameras is not enough to perform robust tracking. Before combining multi-view likelihoods, the peak point of the likelihood function of each view does not provide accurate information about the location of the person. Because of these inaccurate observations the method fails to track people in the scene. In Figure 4.8-a and Figure 4.8-b, we present the tracking errors for each person and the tracking results obtained by our method using 3 coefficients per block, respectively. Clearly, we can see that all people in the scene can be tracked very well by our method. The reason of the peak error value in the third person is because the tracking starts a few frames after the third person enters the room. For this reason, there is a big error at the time third person enters the room. When the number of coefficients taken per block is less then 3, we also observe identity problems. But by selecting the number of coefficients per block greater than or equal to 3, we can track all the people in the scene accurately. The centralized approach, in total, requires at least more than two orders of magnitude coefficients to achieve this level of accuracy. Unlike the decentralized Kalman approach, our compression scheme enables us to decrease the communication and at the same time keep sufficient information to perform robust tracking.

In the light of the results we obtained, we can say that our framework successfully decrease the communication in the network without affecting the tracking performance significantly. For the same tracking performance, our framework saves 99.6% of the

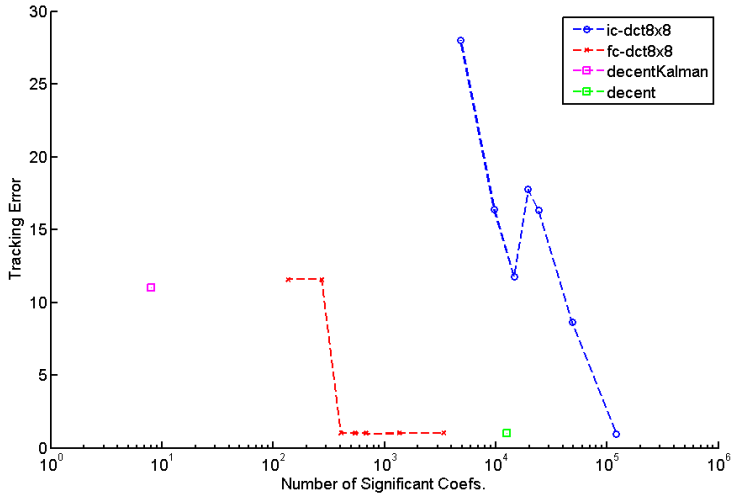


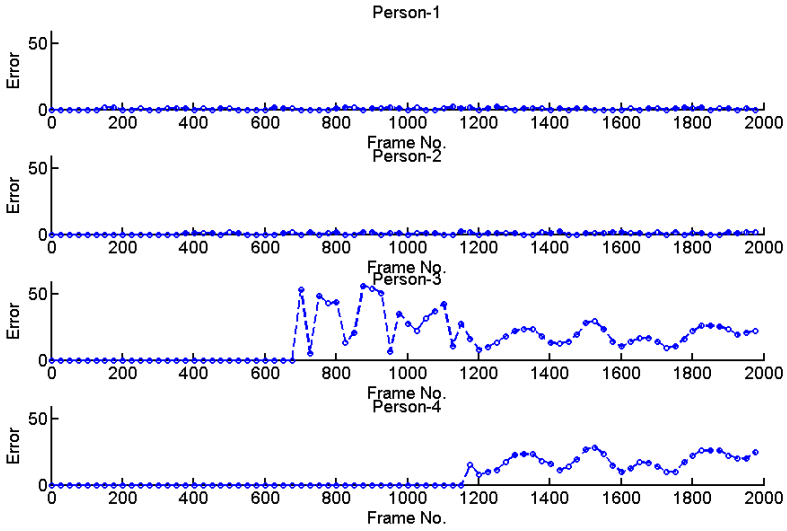

**Figure 4.5**: Indoor sequence: The average tracking errors vs. the number of coefficients for the centralized approach (blue), our framework (red), the decentralized Kalman approach that is similar to the method in [7] (purple) and another decentralized method (green) that directly sends likelihood functions.

bandwidth compared to the centralized approach and achieves saving up to 96.75% compared to the ordinary decentralized approach.

### 4.4.4 Outdoor Tracking Results

The performance of our method for outdoor multi-person tracking is presented in this subsection. Again, we have compared our method with the traditional centralized approach using DCT domain with a block size of 8×8 and the decentralized Kalman approach that is similar to the method in[7]. For our method, we took only the 5, 10, 15, 20, 30 and 50 most significant coefficient(s) per block. Consequently, with the likelihoods of 40×40 size, at each camera in total we end up with at most 125, 250, 375, 500, 750 and 1250 coefficients per person. Since there are four individuals

(a)




(b)

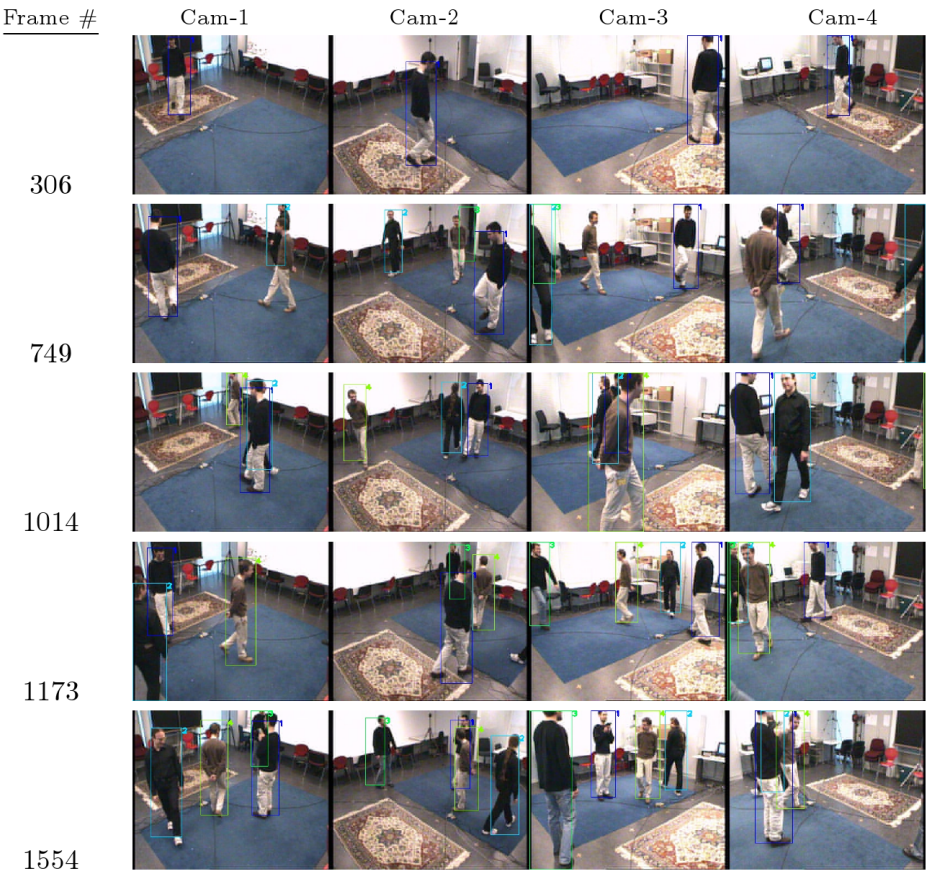
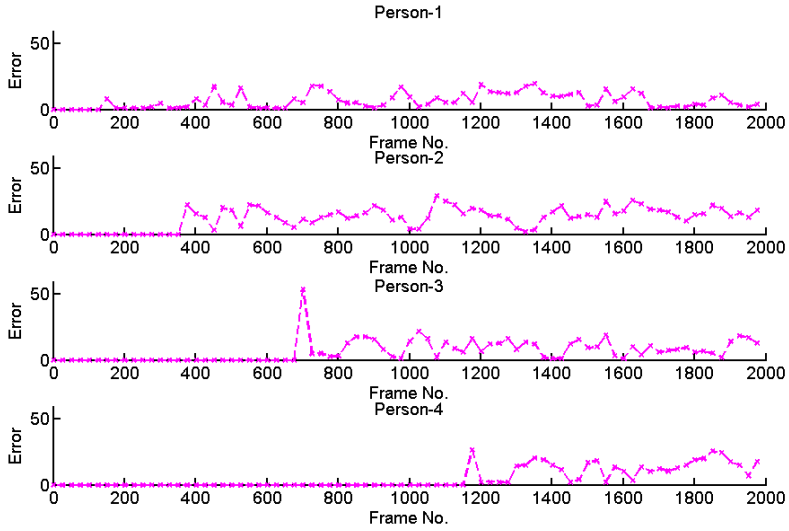



**Figure 4.6**: (a) The tracking errors for each person and (b) tracking results for the indoor dataset obtained by the centralized approach using 48600 coefficients in total in communication.

(a)




(b)

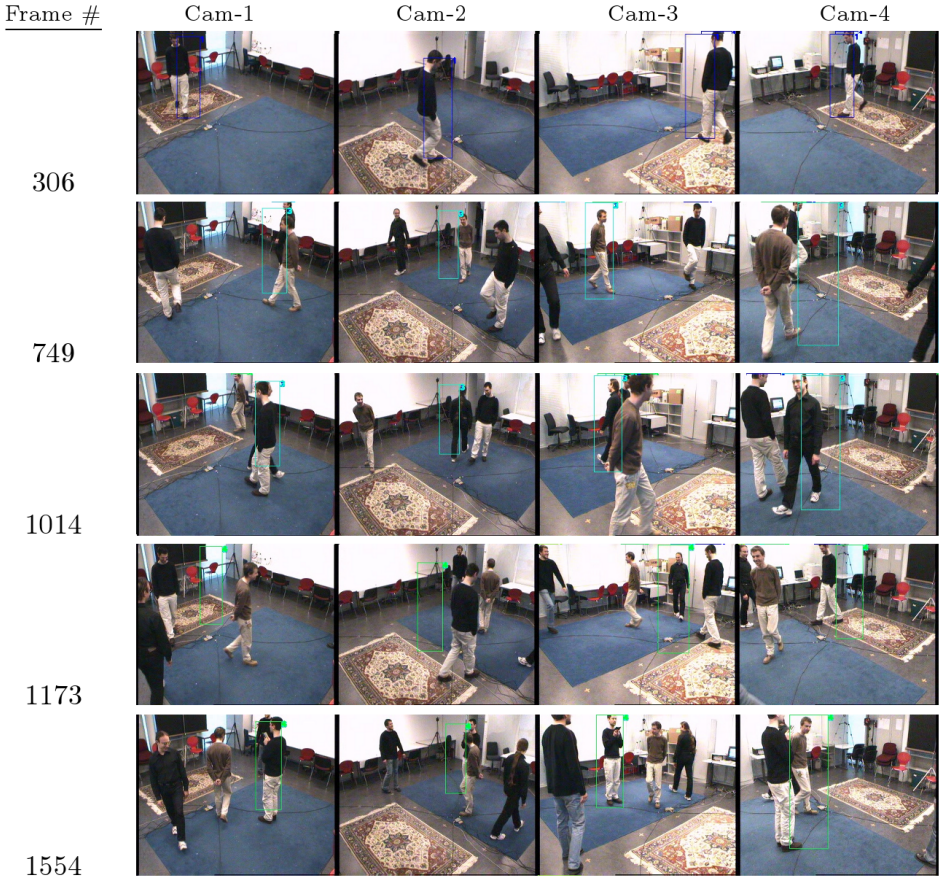
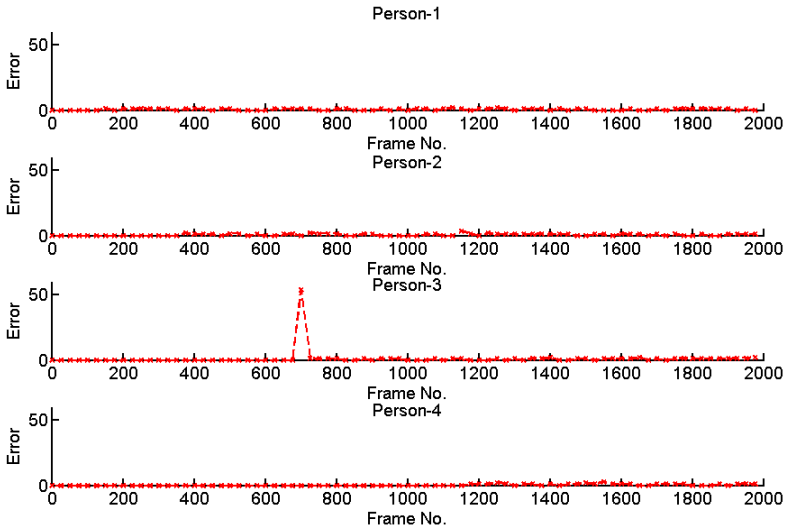
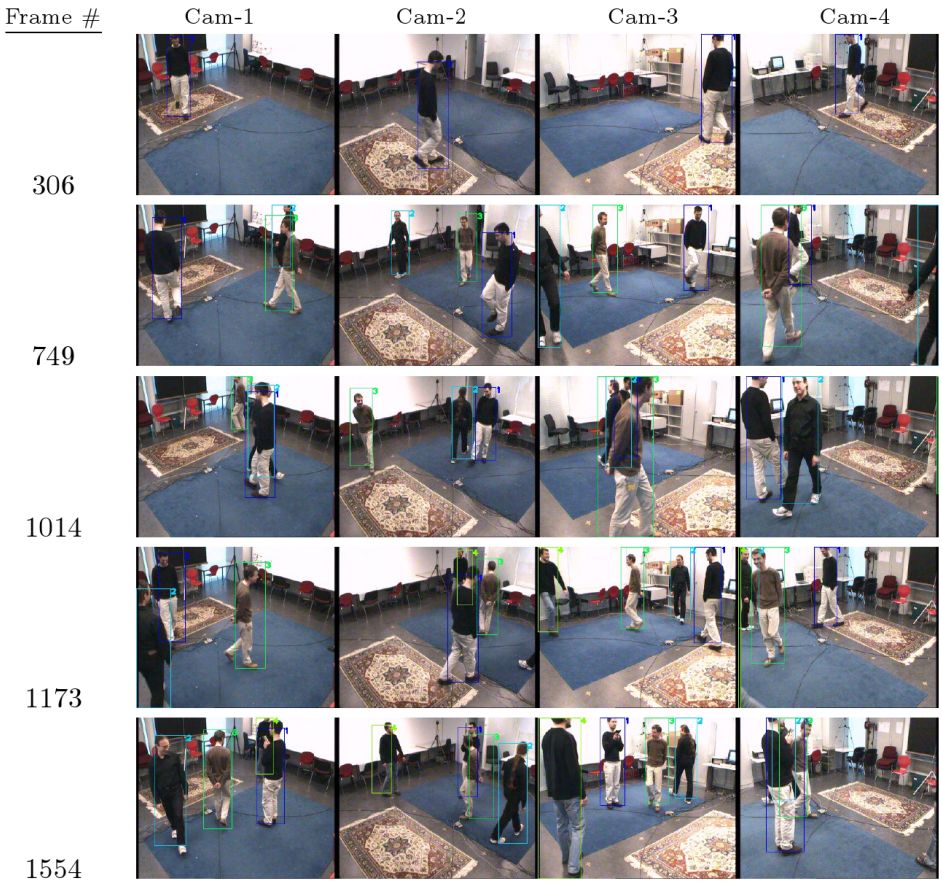



**Figure 4.7**: (a) The tracking errors for each person and (b) tracking results for the indoor dataset obtained by the decentralized Kalman approach.




(a)




(b)

**Figure 4.8**: (a) The tracking errors for each person and (b) tracking results for the indoor dataset obtained by our framework using 3 coefficients per block in communication.

in the scene at maximum, each camera sends at most 500, 1000, 1500, 2000, 3000, and 5000 coefficients to the fusion node. Again, these are the maximum number of coefficients since in some blocks the elements may all be zero, thereby we do not take coefficients from these blocks. In the centralized method, we compress the images by taking only the 1, 2, 3, 4, 5, 10, and 25 most significant coefficient(s) for each block in image. By compressing the images with 360×288 size and 3 color channels, at each camera we end up with 4860, 9720, 14580, 19440, 24300, 48600 and 121500 coefficients. As we have mentioned in Section 4.4.3, in the decentralized Kalman approach, each camera sends only 2 points per person, xy-point of the peak point, to the fusion node. In total, we end up with 8 points in maximum for four individuals.

As in the indoor sequence, tracking errors are evaluated via the Euclidean distance between the tracking and manual marking results. Figure 5.6 presents the average of tracking errors over all people versus the total number of significant coefficients used in communication for our feature compression framework. Again, the maximum number of coefficients is shown here. The performance of the centralized approach is also presented in Figure 5.6. It can be clearly seen that the centralized approach fails to maintain robust tracking while decreasing communication load in the network. It requires at least 121500 significant coefficients in total to achieve an error of around 3 pixels in the grid on average. Using less coefficients with this approach causes identity association problems. On the other hand, by using our framework we achieve an error of 2 pixels in the grid in average with 15 coefficients per block. In our experiments, this led to sending at most 1500 coefficients for four people. Taking less than 15 coefficients per block affects the performance of the tracker and produces an error of 8 pixels in the grid on average. But it can be observed that, our method

significantly outperforms the centralized approach. In Figure 5.6, we also present the performance of the decentralized Kalman approach. We can see that, by using this approach, we can obtain a huge reduction in the communication, but we cannot perform robust tracking. Our framework is also advantageous over an ordinary decentralized approach that directly sends likelihood functions to the fusion node. In such an approach, we send each data point in the likelihood function, resulting a need of sending 6400 values for tracking four people. The performance of this approach is also given in Figure 4.5. It can be seen that we can both achieve the same level of tracking accuracy with the ordinary decentralized method and decrease the communication in the network as well.

The tracking errors for each person and the tracking results, obtained by the centralized approach using 48600 coefficients in total, are given in Figure 4.10-a and Figure 4.10-b, respectively. It can be seen that although the centralized approach can track the second and the fourth individuals very well, there is an identity association problem for the third, fifth individuals and at the end of first person. Figure 4.11 shows the tracking errors of each person and tracking results obtained by the decentralized Kalman approach. It can be seen that it fails to track the people in the scene. Nearly for all people, there occurs identity association problems. Usually, it loses the track of the person and starts tracking a virtual person in the scene (Figure 4.7-b). As in Section 4.4.3, the reason of these failures are that the amount of information coming from cameras is not enough to perform robust tracking. The peak point of the likelihood function of each view does not provide accurate information about the location of the person. Thus, the method fails to track people in the scene. In Figure 4.12-a and Figure 4.12-b, we present the tracking errors for each person and the tracking re-

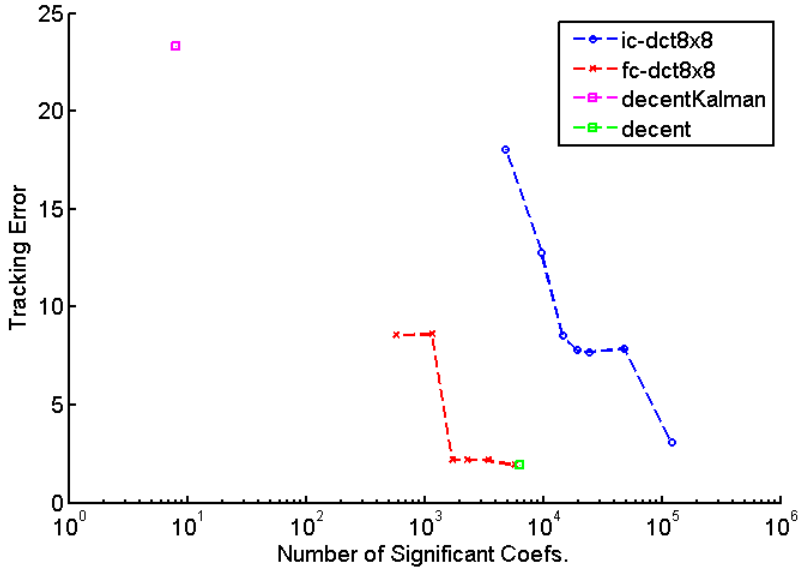
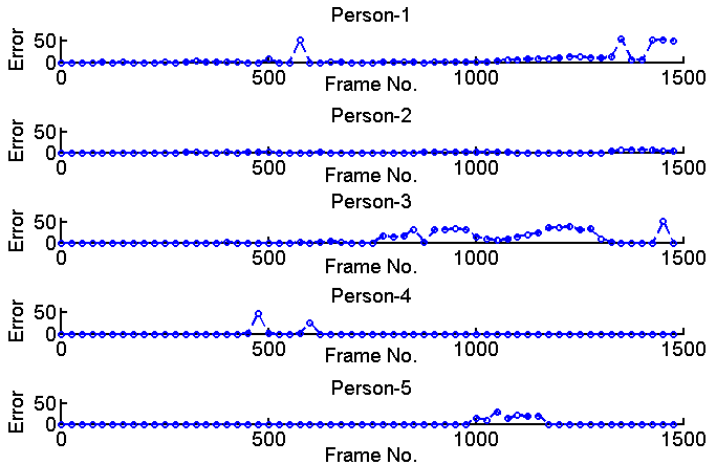


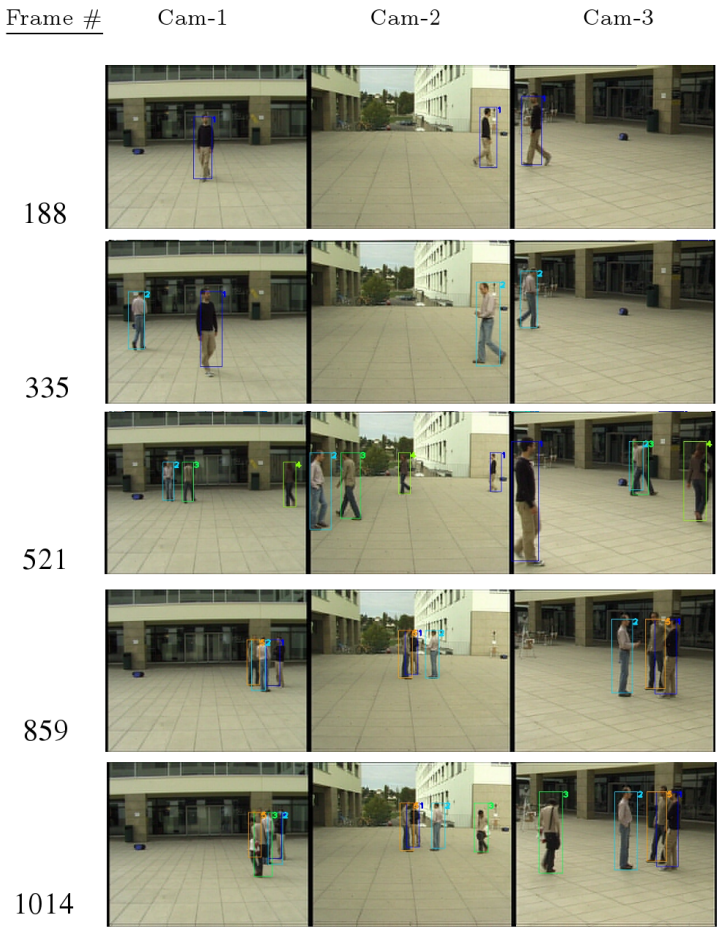

**Figure 4.9**: Outdoor sequence: The average tracking errors vs. the number of coefficients for the centralized approach (blue), our framework (red), the decentralized Kalman approach that is similar to the method in [7] (purple) and another decentralized method (green) that directly sends likelihood functions.

sults obtained with our method using 15 coefficients per block, respectively. Clearly, we can see that all people in the scene can be tracked very well by our method. The reason of the peak error value in the first, third and fourth person is that the people leaves or enters the scene. Either the tracking starts a few frames after the person enters the scene or stops a few frames before the person leaves the scene. For this reason, there is a big error at these time instances. When the number of coefficients taken per block are less then 15, we also observe identity problems. But by selecting the number of coefficients per block greater than or equal to 15, we can track all the people in the scene accurately. The centralized approach, in total, requires almost two orders of magnitude coefficients to achieve this level of accuracy.

Based on these results, we can say that our framework successfully decreases the communication in the network without affecting the tracking performance significantly. For the same tracking performance, our framework saves 98.7% of the bandwidth compared to the centralized approach and achieves 76.5% saving compared to the decentralized approach.

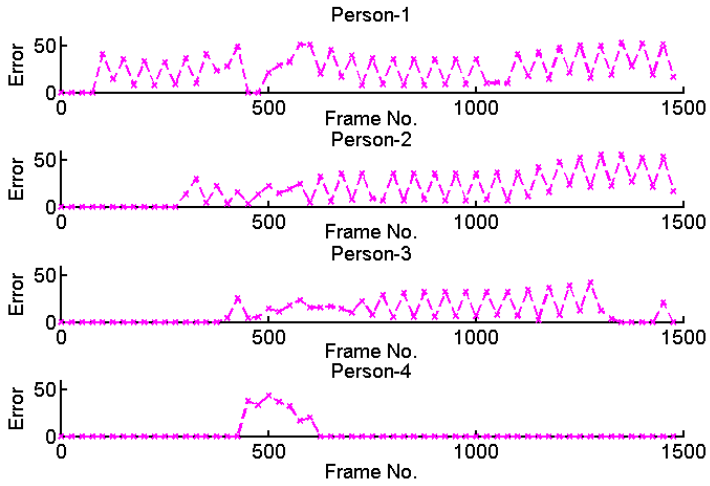



(a)

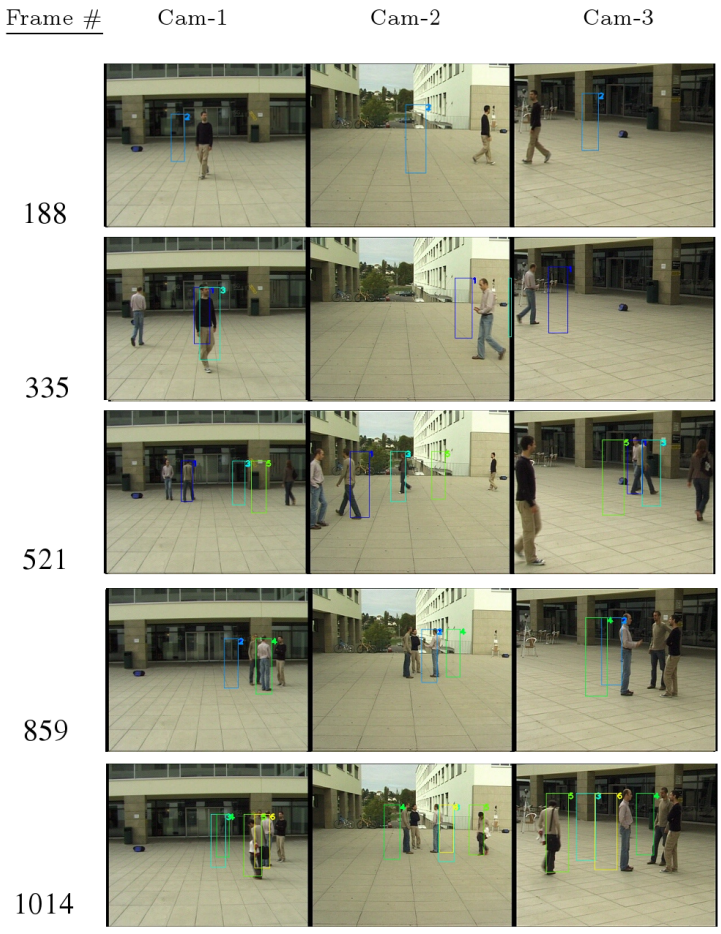



(b)

**Figure 4.10**: (a) The tracking errors for each person and (b) tracking results for the outdoor dataset obtained by the centralized approach using 48600 coefficients in total in communication.

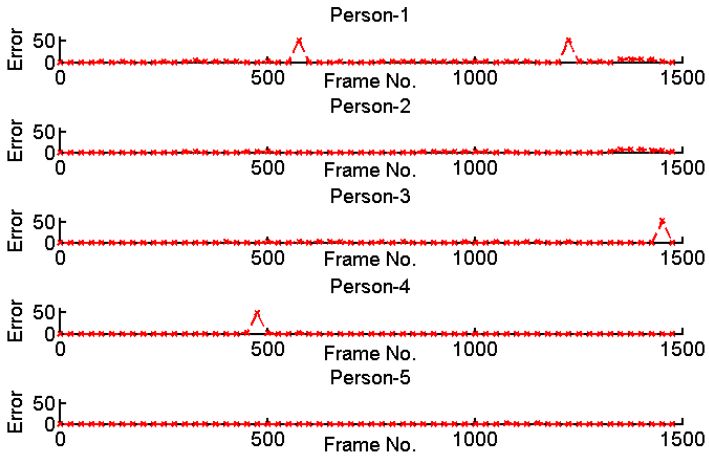



(a)

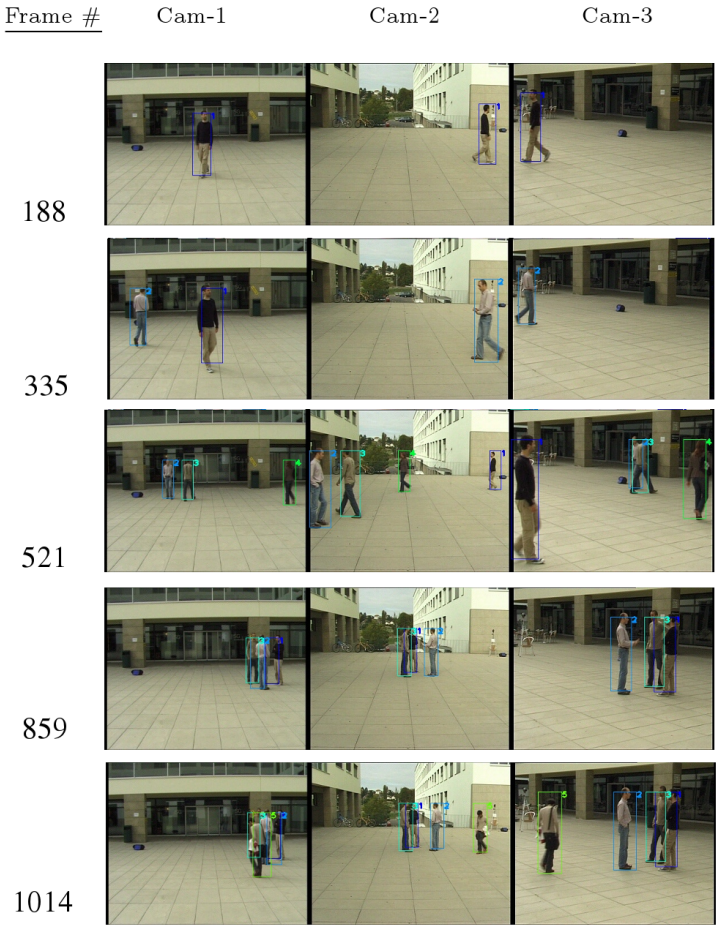



(b)

**Figure 4.11**: (a) The tracking errors for each person and (b) tracking results for the outdoor dataset obtained by the decentralized Kalman approach.




(a)




(b)

**Figure 4.12**: (a) The tracking errors for each person and (b) tracking results for the outdoor dataset obtained by our framework using 15 coefficients per block in communication.

# 5 A SPARSE REPRESENTATION FRAMEWORK FOR HUMAN TRACKING IN VSNS

This chapter presents our sparse representation framework for human tracking in visual sensor networks. Although this framework attacks the same problem with the framework in Chapter 4, this is a different approach that uses sparse representation tools and is another contribution of this thesis. As we have explained in Chapter 4, decentralized approaches fit well to VSNs. Rather than a block-based likelihood compression scheme, here we design special overcomplete dictionaries that are matched to the structure of the likelihood functions and use these dictionaries for sparse representation of likelihoods. By using these dictionaries, we can represent likelihoods with few coefficients, and thereby decrease the communication between cameras and fusion nodes. In Section 5.1, we describe how we design the dictionaries. To the best of our knowledge, this is the first method in which likelihood functions computed in the context of tracking in a VSN are compressed by sparse representation.

As we have seen in Section 2.1.2, the implementation of solvers for sparse representation is an important step in real-world applications. Based on this observation, we have performed a bench-marking analysis on $l_1$ solvers in order to find the best

method in time and accuracy for our sparse representation framework.

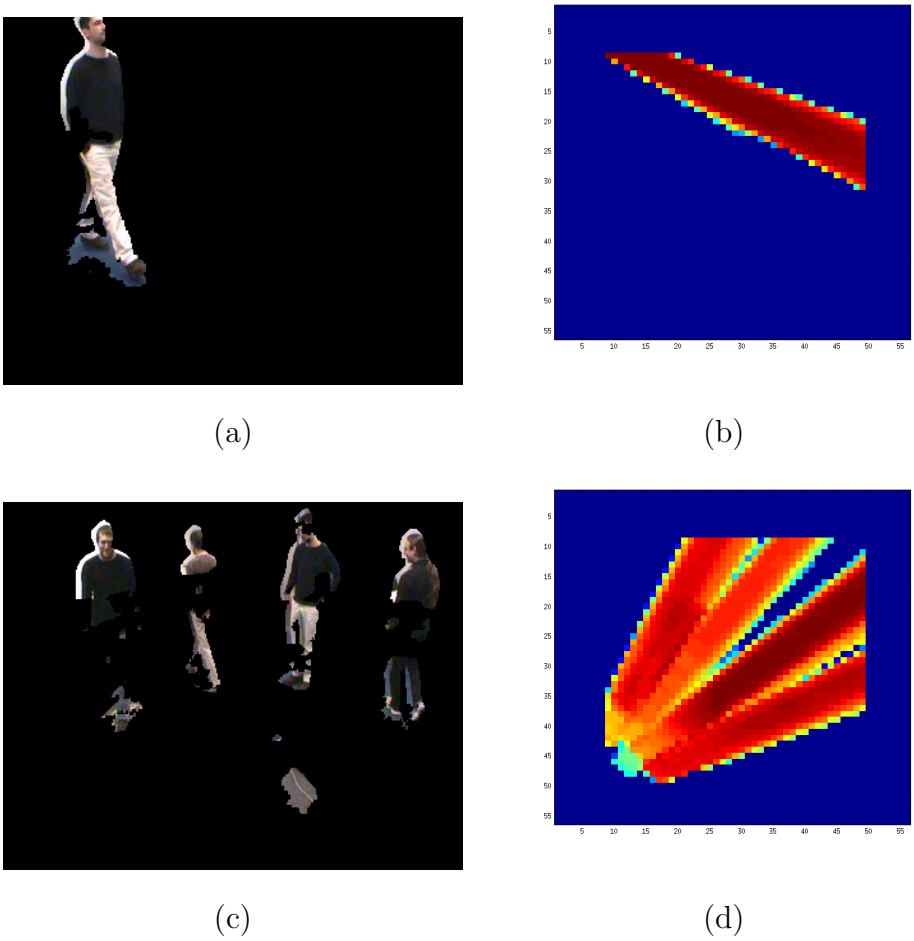
The performance of our sparse representation framework is presented in Section 5.3 by comparing with our framework in Chapter 4 and a distributed tracking algorithm [9] via both qualitative and quantitative results.

## 5.1 Sparse Representation of Likelihoods

In Chapter 4, we have shown that likelihood compression through orthogonal transforms of blocks of the likelihood functions can achieve bandwidth reduction. Also, we have seen that there are some limitations of this framework. After a certain level of bandwidth reduction, we cannot reduce further without decreasing the tracking performance (Section 4.4). The reason is that this approach does not take into account the structure of the likelihood functions. Although such transforms provide some level of sparsity, they certainly do not fully exploit the structure of the likelihood functions. For computational reasons, these orthogonal transforms are applied to blocks of the likelihood functions, leading to blocking artifacts in the reconstructed likelihoods. Most of the wavelet transforms try to extract the edge regions in the image. But the likelihood functions have rarely edge regions. Focusing on scenarios involving extreme bandwidth constraints, we believe that we can achieve much more bandwidth reduction by representing likelihood functions sparsely in an appropriate domain. Considering the fact that the domain is the key component that determines the level of sparsity, thereby the level of reconstruction accuracy, we need a special dictionary that is matched with the structure of the full likelihood functions.

In fact, the likelihood functions we obtain from the color model in [6] have a special structure. As it has been explained in Section 4.1.2, the color model likelihood functions for a person of interest are obtained by comparing the color histogram of rectangular patches in foreground image and a template color histogram of the person of interest. In Figure 5.1, two sample foreground images and the likelihood functions obtained from these foreground images are given. Figure 5.1-a and Figure 5.1-c show foreground images captured from two different camera views when there is only one person in the scene and when the scene is crowded, respectively. The likelihood function obtained from these image are given in Figure 5.1-b and Figure 5.1-d. We can clearly see that likelihood functions consist of quadrilateral-shaped components. A person in the scene creates a quadrilateral-shaped component in the likelihood function. One of the important properties of these components is that their shape do not depend on the value of the foreground pixels. The values inside the quadrilateral changes according to the color pixels in foreground image. But the shape of the trapezoid only depends on the camera view and the position of the foreground pixels. For this reason, we can say that these quadrilateral-shaped components are building blocks of likelihood functions. By creating an overcomplete dictionary from these building blocks, we can naturally and properly utilize the structure of the whole likelihood functions.

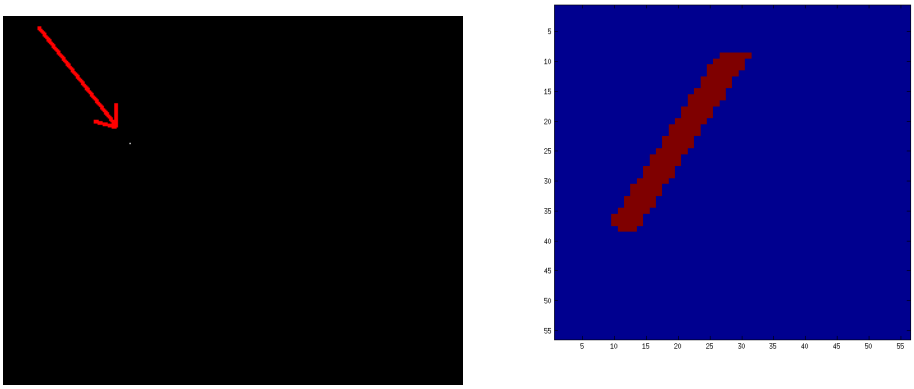
In Section 4.3, we have discussed about parameters that measure the abilities of wavelet transforms in frequency and time domain for representing functions. Similarly, while creating this dictionary, it is very important to cover all possible types of likelihood functions. In order to find all the building blocks of likelihood functions, we need to create likelihood functions from all possible foreground images. We start



(a) (b)

(c) (d)

**Figure 5.1**: Foreground images captured from two different camera views (a) when there is only one person in the scene, (c) when the scene is crowded and (b,d) color model likelihood functions obtained from the images .



(a) (b)

**Figure 5.2**: (a) A sample foreground image that is all-black except a white pixel (pointed with a red arrow) and (b) the likelihood function obtained from this foreground.

by a foreground image that is all-black except a single white pixel, which can be described as a building block of foreground images. By using a flat template histogram, we obtain a likelihood function from this image. We assume that, since it is obtained from the building block of foreground images, this likelihood function is one of the building blocks of likelihood functions (Figure 5.2). By changing the position of the white pixel and re-evaluating the likelihood function from that foreground image, we can create a pool composed of building blocks. For each camera, we create the dictionary matrix ($A_c$ in Eq. 5.1) by arranging the building blocks of likelihoods as the columns of the matrix.

Using these overcomplete dictionaries, we can build the following sparse representation framework:

$$\min_{x_c} ||y_c - A_c \cdot x_c||_2 + \lambda ||x_c||_1 \tag{5.1}$$

where $y_c$ and $x_c$ represents the likelihood function of the camera $c$ (for a person of interest in a particular time instant) and its sparse coefficients, respectively, and $A_c$ is the dictionary that we have created for camera $c$.

To obtain the sparse representation of the likelihood function, at each camera we solve the optimization problem in Eq. 5.1. Then, each camera sends this sparse representation to fusion node. At the fusion node, likelihood functions of each camera can be reconstructed simply by multiplying the new representation with the matrix $A_c$ and used in tracking.

As in our feature compression framework, our sparse representation framework does not require the use of specific image features or likelihood functions.

## 5.2 Comparison of Solvers for $l_1$-minimization

As we have seen in Section 2.1.2, solving optimization problems with $l_1$ constraint has become a well-established research area. Many solver algorithms has been proposed for $l_1$-minimization. In [109], the solvers we have described in Section 2.1.2 have been compared in order to find the fastest solver to be used in sparsity-driven face recognition. Following this work, we have compared these solvers in order to choose a solver that is fast and accurate for our sparse representation framework.

In the comparison, we have used 20 different likelihood functions obtained from the tracker in [6] as a test set. Our special design dictionaries, that have been created using the approach in the previous section, have been used as the $A_c$ matrix in Eq. 5.1.

| | $\lambda = 0.1$ | $\lambda = 0.5$ | $\lambda = 1$ | $\lambda = 10$ | $\lambda = 100$ | $\lambda = 200$ |
|---|---|---|---|---|---|---|
| Homotopy [107] | 0.60133 | 0.64638 | 0.62925 | 0.62143 | 0.49892 | 0.32343 |
| L1_LS [38] | 816.1898 | 134.1769 | 81.6986 | 30.0675 | 13.7835 | 12.8253 |
| SpaRSA [47] | 24.695 | 22.9193 | 22.1939 | 21.7195 | 22.0352 | 0.063627 |
| FISTA [108] | 636.0806 | 381.3541 | 343.2617 | 206.3525 | 130.385 | 131.4214 |
| ALM [109] | 0.085406 | 0.078551 | 0.079087 | 0.078963 | 6.8105 | 127.8755 |

**Table 5.1**: Average run-times of solvers in seconds for different regularization parameters.

Homotopy [107], L1LS [38], SpaRSA [47], FISTA [108] and ALM [109] algorithms have been used to solve the optimization problem in Eq. 5.1 and find the sparse representation of likelihood functions in the test set. The regularization parameter in Eq. 5.1, $\lambda$, is set as $0.1, 0.5, 1, 10, 100, 200$. The average run-times of the solvers in seconds for different $\lambda$ values has been shown in Table 5.1. To avoid trivial solutions, we have also checked the number of iterations of each algorithm . The average iteration count solvers for different $\lambda$ values are given in Table 5.2.

We can see that ALM algorithm for $\lambda$ values between $0.1 - 10$ and SpaRSA algorithm for $\lambda = 200$ are the fastest solvers. But, average iteration counts of ALM and SpaRSA show that they find the trivial solution. When we look at the Homotopy algorithm, we can see that, independent of the $\lambda$ parameter, Homotopy algorithm works both fast and accurate. We have observed that selecting $\lambda = 0.1$ enforces enough level of sparsity to achieve reasonable results. Therefore, we select the Homotopy algorithm and set $\lambda = 0.1$ in our tracking experiments.

| | $\lambda = 0.1$ | $\lambda = 0.5$ | $\lambda = 1$ | $\lambda = 10$ | $\lambda = 100$ | $\lambda = 200$ |
|---|---|---|---|---|---|---|
| Homotopy [107] | 15.4792 | 15.4792 | 15.4792 | 15.1042 | 8.6354 | 1.3125 |
| L1_LS [38] | 20.8333 | 17.9479 | 18.8542 | 21.1979 | 20.5104 | 21.5833 |
| SpaRSA [47] | 833.5938 | 848.9271 | 833.5729 | 813.4583 | 826.0208 | 0.5 |
| FISTA [108] | 269.9688 | 144.4583 | 121.9375 | 47.7917 | 7.8542 | 4.75 |
| ALM [109] | 2 | 2 | 2 | 2 | 206.0313 | 3498.6563 |

**Table 5.2**: Average iteration count of solvers for different regularization parameters.

## 5.3 Experimental Results

### 5.3.1 Comparison with the Block-based Compression Framework

In this subsection, we have presented experimental results that we obtained using the tracking framework in [6] (Section 4.1.2). We have compared our method with the block-based compression framework in Chapter 4.

#### Setup

In the experiments, as we did in Section 4.4, we have simulated the VSN environment by using the indoor and outdoor multi-camera dataset in [6].

Using the procedure explained in Section 5.1, we have created the dictionaries for each view. Since the foreground images are in size of 360x288, we obtain 3 and 4 overcomplete dictionaries with 103680 atoms for the outdoor and indoor datasets respectively. We have observed that, because of the overlap between rectangular patch and white pixel in the foreground image, some atoms are all-zero or same with other atoms. As a post-processing step, we remove these atoms from the matrix. For

the indoor dataset, we end-up with dictionaries with 36073, 46986, 28155 and 30195 atoms for first, second, third and fourth view, respectively. For the outdoor dataset, we end-up with dictionaries with 12777, 11984, and 19846 atoms for first, second, and third view, respectively.

Following our observations in previous section, we have solved the optimization problem using the Homotopy algorithm [107] with $\lambda$ is set as 0.1 for all dictionaries.

### Indoor Tracking Results

In this subsection, we present the performance of our method used for indoor multi-person tracking. In the experiments, we have compared our method with the block-based compression framework in Chapter 4. Here we consider the version of the framework that uses DCT for feature compression with a block size of 8×8 and took only the 1, 2, 3, 4, 5, 10, and 25 most significant coefficient(s) per block. Consequently, with the likelihoods of 56×56 size, at each camera in total we end up with at most 49, 98, 147, 196, 245, 490 and 1225 coefficients per person. Since there are four individuals in the scene at maximum, each camera sends at most 196, 392, 588, 784, 980, 1960 and 4900 coefficients. As we have mentioned in Section 4.4, according to the structure of the likelihood functions, the elements in a block may all be zero. For this reason, these are the maximum number of coefficients. In our method, after sparse representation of color model likelihood of a person of interest is found, we only took 10, 15, 20, 25, 50 and 100 most significant coefficients. This provides that, since there are four individuals in the scene at maximum, each camera sends at most 40, 60, 80, 100, 200 and 400 coefficients to the fusion node.

As we have stated in Section 4.4, a groundtruth for this sequence is obtained by manually marking the people in the ground plane, in intervals of 25 frames. Tracking errors are evaluated via Euclidean distance between the tracking and manual marking results (in intervals of 25 frames). Figure 5.3 presents the average of tracking errors over all people versus the total number of significant coefficients used in communication for our sparse representation framework. Since the total number of significant coefficients sent by a camera may change depending on the number of people at that moment, the maximum is shown in Figure 5.3. The performance of our block-based compression framework is also presented in Figure 5.3. It can be clearly seen that by using the custom-designed dictionaries, our sparse representation framework achieves much more bandwidth reduction than using the block-based compression framework. To achieve an error of 1 pixel in the grid in average, our sparse representation framework using custom-designed dictionaries needs at least 20 coefficients per person, whereas the block-based compression framework needs at least 147 coefficients per person.

The tracking results of the block-based compression framework using 49 coefficients per person and our sparse representation framework using 20 coefficients per person are given in Figure 5.4 and Figure 5.5, respectively. It can be seen that, although the block-based compression approach can track the first and the second individuals very well, there is an identity association problem for the third and fourth individuals. Clearly, we can see that all people in the scene can be tracked very well by our sparse representation framework. The reason of the peak error value in the third person is because the tracking starts a few frames after the third person enters the room. For this reason, there is a big error at the time third person enters the room.

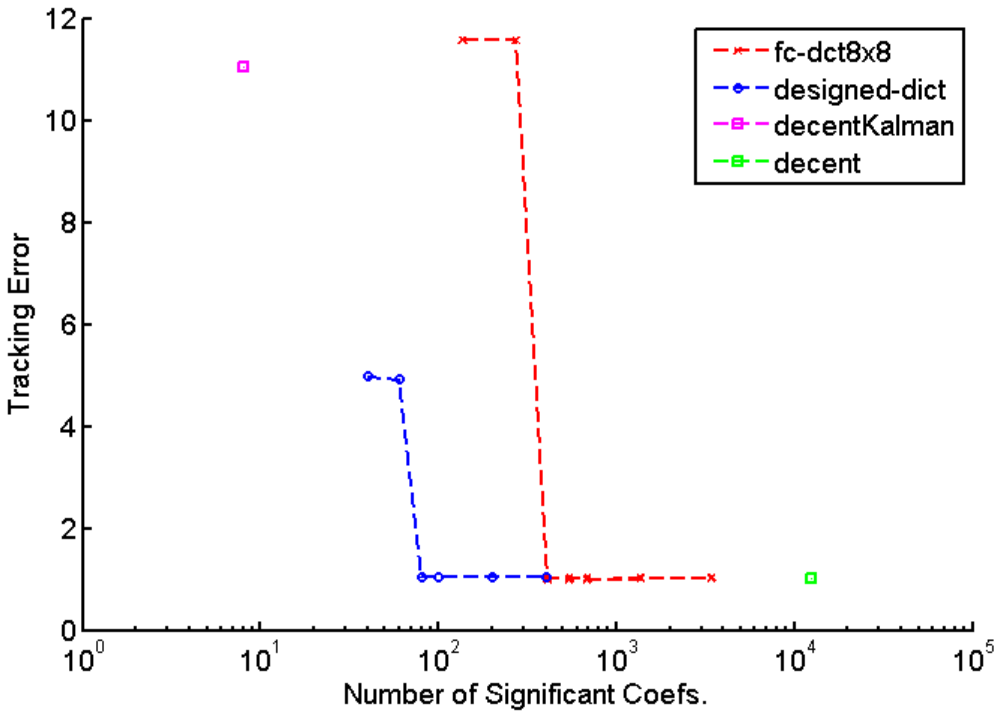



**Figure 5.3**: Indoor sequence: The average tracking errors vs. the number of coefficients for our block-based compression framework in Chapter 4 (red), our sparse representation framework (blue) and a decentralized method (green) that directly sends likelihood functions.

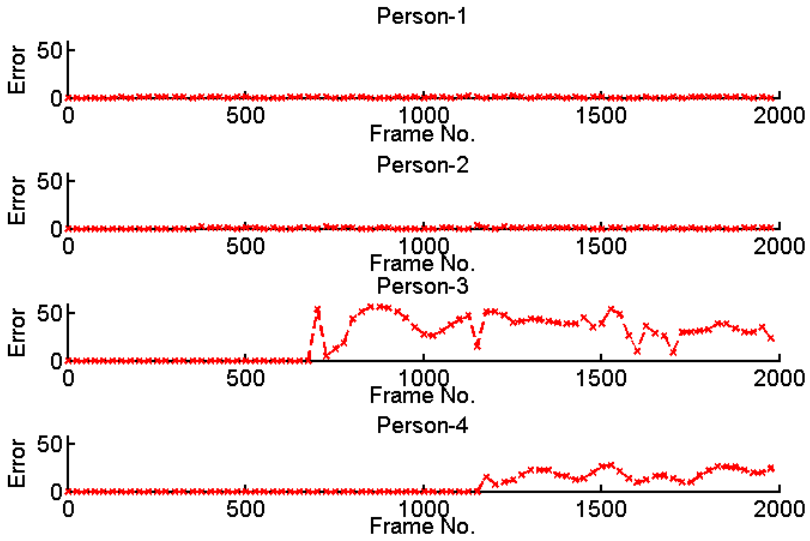
When the number of coefficients taken per person are less then 20, we also observe identity problems. But by selecting the number of coefficients per person greater than or equal to 20, we can track all the people in the scene accurately. The block-based compression framework, in total, requires at least five times more coefficients to achieve this level of accuracy.

In the light of the results we obtained, for the same tracking performance, our sparse representation based method saves 80.39% of the bandwidth used by the block-based compression approach. Our method representation based is also advantageous over an ordinary decentralized approach that directly sends likelihood functions to the fusion node. In such an approach, we send each data point in the likelihood function, resulting a need of sending 12544 values for tracking four people. The performance of this approach is also given in Figure 5.3. Our approach uses only 1.25% of the bandwidth needed by the decentralized approach.

**Outdoor Tracking Results**

The performance of our sparse representation based method for outdoor multi-person tracking is presented in this subsection. Again, we have compared our sparse representation framework with the block-based compression framework in Chapter 4 using DCT domain with a block size of 8×8. For this approach, we took only the 5, 10, 15, 20, 30, and 50 most significant coefficient(s) per block. Consequently, with the likelihoods of 40×40 size, at each camera in total we end up with at most 125, 250, 375, 500, 750 and 1250 coefficients per person. Since there are four individuals in the scene at maximum, each camera sends at most 500, 1000, 1500, 2000, 3000, and 5000 coefficients. Again, these are the maximum number of coefficients since the elements

(a)




(b)

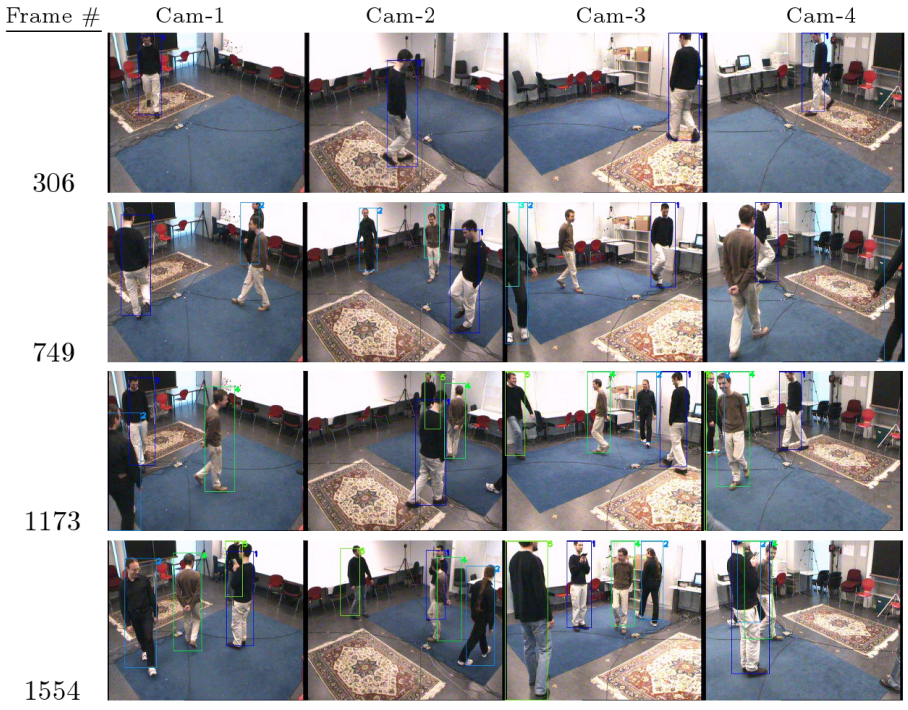
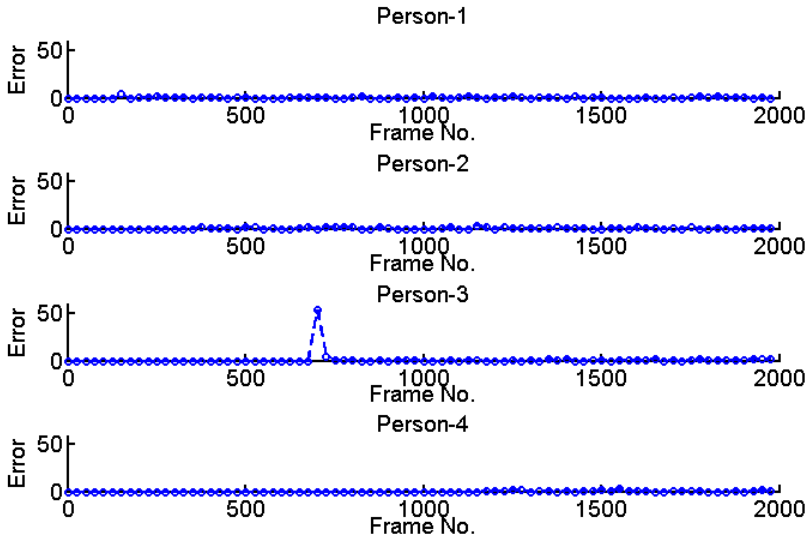



**Figure 5.4**: (a) The tracking errors for each person and (b) tracking results for the indoor dataset obtained by the block-based compression framework in Chapter 4 using 49 coefficients per person used in communication.

(a)




(b)

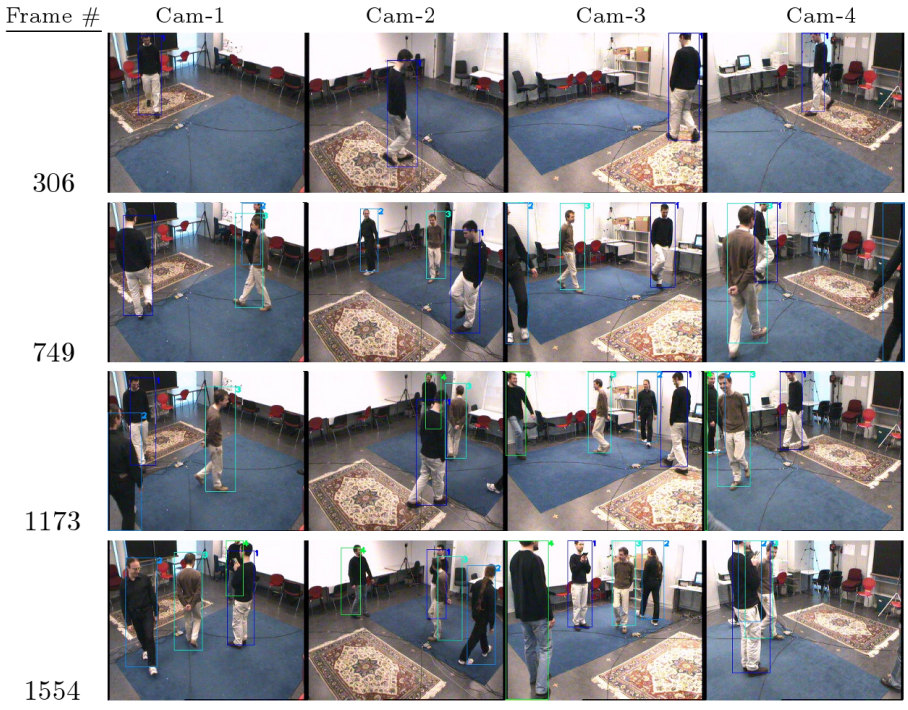



**Figure 5.5**: (a) The tracking errors for each person and (b) tracking results for the indoor dataset obtained by our sparse representation framework using 20 coefficients per person used in communication.

in a block may all be zero. In our method, after sparse representation of color model likelihood of a person of interest is found, we only took 5, 10, 15, 20, 25, 50 and 100 the most significant coefficients. This provides that, since there are four individuals in the scene at maximum, each camera sends at most 20, 40, 60, 80, 100, 200 and 400 coefficients to the fusion node.

As in the indoor sequence, tracking errors are evaluated via the Euclidean distance between the tracking and manual marking results. Figure 5.6 presents the average of tracking errors over all people versus the total number of significant coefficients used in communication for our sparse representation framework. Again, the maximum number of coefficients is shown here. The performance of our block-based compression framework is also presented in Figure 5.6. It can be clearly seen that our sparse representation framework works better in decreasing the communication than the block-based compression framework. The block-based compression framework requires at least 375 coefficients per person to achieve an error of 2 pixel in the grid in the average. Using less coefficients with this approach causes identity association problems. On the other hand, by using our sparse representation framework we achieve same tracking performance with 10 coefficients per person.

The tracking results of the block-based compression framework using 250 coefficients per person and our sparse representation based approach with custom-designed dictionaries using 10 coefficients per person are given in Figure 5.7 and Figure 5.8, respectively. It can be seen that, the block-based compression fails to preserve identities. Especially, when a person leaves the scene and comes back, the person cannot be recognized and he or she is considered as a new person in the scene. Clearly, we

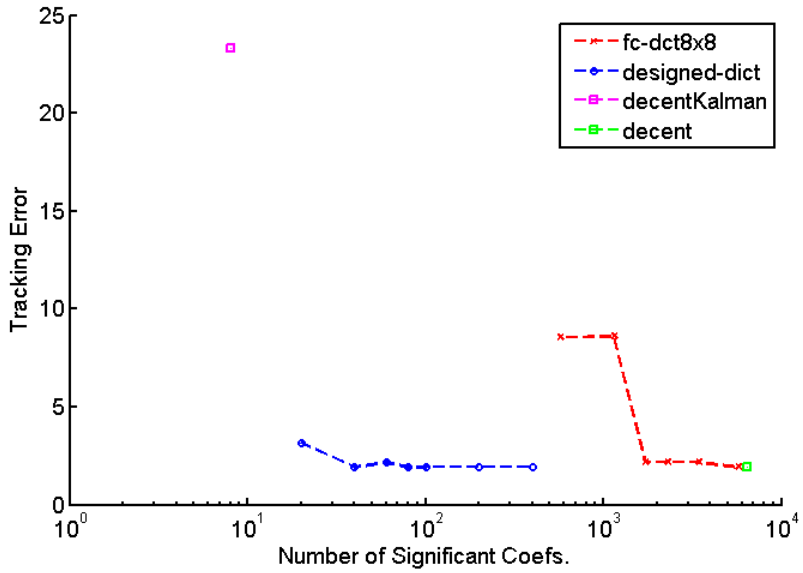



**Figure 5.6**: Outdoor sequence: The average tracking errors vs. the number of coefficients for our block-based compression framework in Chapter 4 (red), our sparse representation framework (blue) and a decentralized method (green) that directly sends likelihood functions.

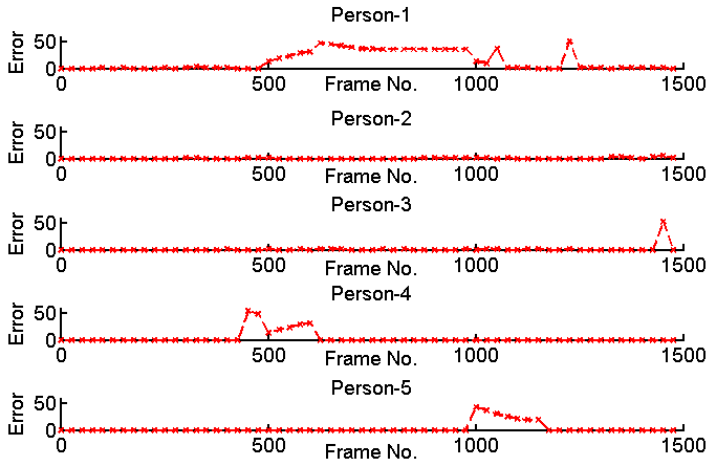
can see that all people in the scene can be tracked very well by our approach with custom-designed dictionaries using 30 times less coefficients.

Based on these results, we can say that, by using the special design dictionaries, our sparse representation framework successfully decreases communication load in the network without significantly degrading tracking performance. It also works better than the block-based compression framework in Chapter 4. Our sparse representation based method is also advantageous over an ordinary decentralized approach that sends 6400 values for tracking four people (Figure 5.6). Our approach uses only 0.63% of the bandwidth needed by the decentralized approach.

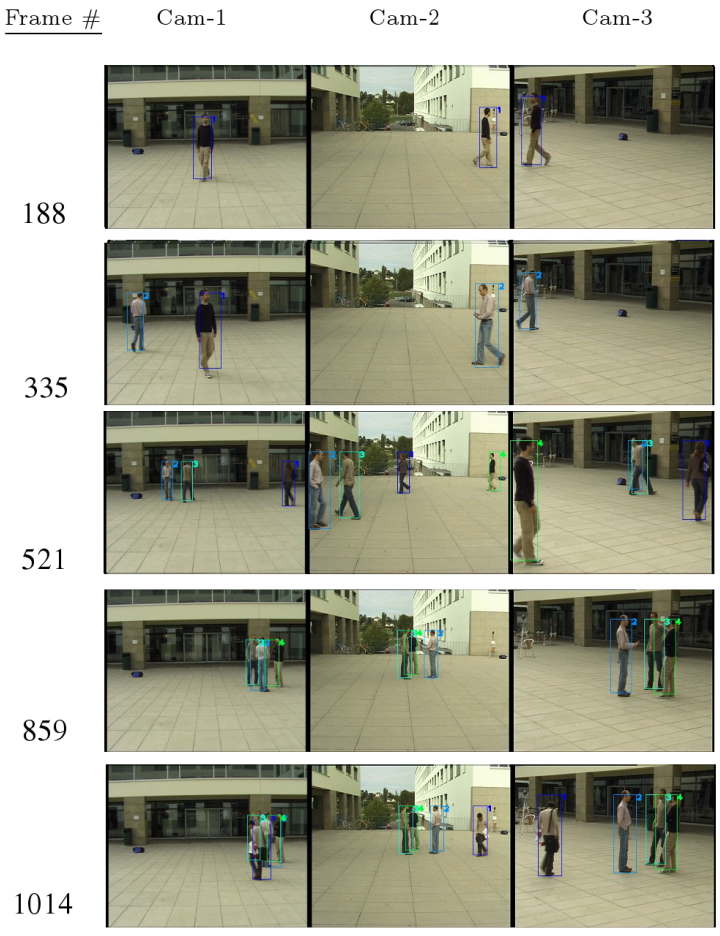
### 5.3.2 Comparison with a Distributed Approach

In this subsection, we present the results of comparing our method with a distributed approach in which each camera node fuses its observations with tracking results received from its neighbors and sends the updated estimates to the next neighbor. We have used the distributed tracking method in [9] for the comparison.

In [9], as we have mentioned in Section 2.2.3, at each camera node the position of people on ground plane is estimated by applying individual Kalman-consensus filters [95] on its own observations together with observations and estimates coming from neighboring cameras. The state of each person in the Kalman-consensus filter is a four-element vector representing the position and velocity on x and y axes on ground plane. We obtain the observation of each camera by using the human detection algorithm in [126]. The human detector outputs a probability map that represents the probable locations of people on image plane. As in [77], we project this

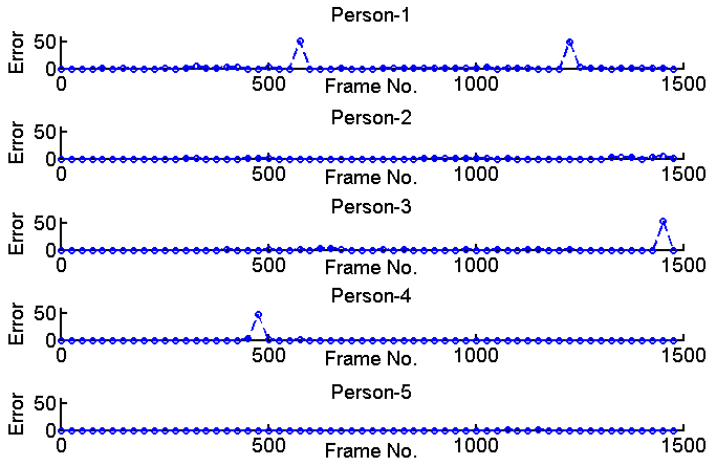



(a)

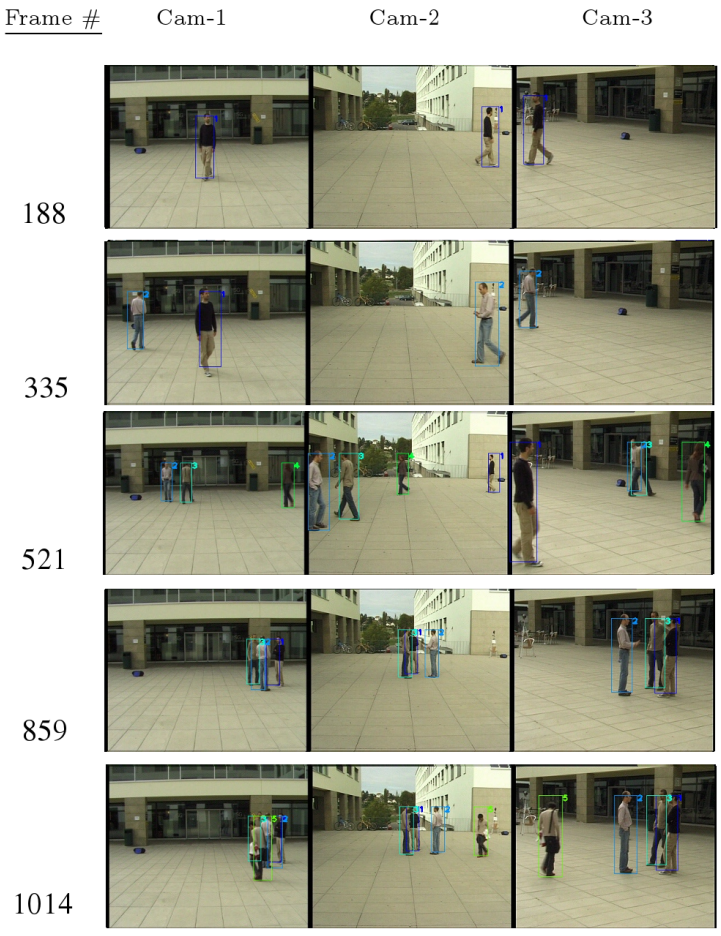



(b)

**Figure 5.7**: (a) The tracking errors for each person and (b) tracking results for the outdoor dataset obtained by the block-based compression framework in Chapter 4 using 10 coefficients per block in communication.




(a)




(b)

**Figure 5.8**: (a) The tracking errors for each person and (b) tracking results for the outdoor dataset obtained by our sparse representation framework using 10 coefficients per person in communication.

probability map onto the ground plane (Z=0) and onto planes at different heights (Z=$200, 400, \cdots, 1600$) that are parallel to the ground. Then, we combine these multi-layered projected probability maps and obtain a likelihood function for a camera view. The observation vector of each camera that is used in Kalman-consensus filter, is obtained by finding the local maximum points of its likelihood function. At each time step, camera nodes share their observation vector and observation covariance together with predicted states of each person. As in [9], the association of observations to people is achieved based upon appearance (color) and motion information.

In our framework, at each camera node we find the sparse representation of the likelihood function, that is obtained by combining multi-layered projected probability maps, using Eq. 5.1 and send this representation to the fusion node. As we have described in Section 5.1, we design a dictionary for each camera by using the building blocks of the likelihood functions. For this tracking method, we create the building blocks by setting a $100 \times 30$ rectangular patch in an all-zero probability map '1' and projecting this map onto ground plane. Similar to the procedure in Section 5.1, as we shift this rectangular patch, we can create a pool of building blocks and consequently build the dictionary. In the fusion node, the likelihoods are reconstructed using the the sparse representation sent by cameras. As we explained in previous chapters, the posterior probability is obtained by combining the likelihood with a motion model and the position of people are estimated by running dynamic programming on the posterior probability.


**Figure 5.9**: A sample set of images from PETS 2009 benchmark dataset [8].

### Setup

In the experiments, again we have simulated the VSN environment by using the PETS 2009 benchmark dataset [8]. The dataset was shot in a university campus and it includes many people appearing simultaneously. We have used the four cameras that cover a rectangular region on ground. The area of interest at this sequence is of size 6 $m \times 6$ $m$ and discretized into $G = 40 \times 40 = 1600$ locations, corresponding to a regular grid with a resolution of 15 $cm$.

The calibration parameters for each camera are provided within the dataset. The size of the images are $720 \times 576$ pixels and the frame rate for all of the cameras is 7 fps. A sample set of images for the dataset is shown in Figure 5.9.

We have created the dictionaries for each view using the procedure explained above. We obtain dictionaries with 6932, 7870, 7768 and 6844 atoms for first, second, third and fourth view, respectively. Again following our observations in previous section, we have solved the optimization problem using the Homotopy algorithm [107] with $\lambda$ is set as 0.1 for all dictionaries.

### Tracking Results

The results of the comparison between our method and the distributed method in [9] is presented in this subsection. In the distributed approach, each camera shares the observation vector (2 element vector) and observation covariance (2×2 matrix) together with the predicted states (4 element vector) with neighboring cameras. In total, 10 elements per person is shared among cameras. Since there are four individuals in the scene at maximum, each camera sends at most 40 elements. In our method, after sparse representation of likelihood of a person of interest is found, we only took 30, 40, 50, 75, 100 most significant coefficients. This provides that, since there are four individuals in the scene at maximum, each camera sends at most 120, 160, 200, 300, and 400 coefficients to the fusion node.

A groundtruth for this sequence is obtained by manually marking the people in the ground plane. Tracking errors are evaluated via Euclidean distance between the tracking and manual marking results. Figure 5.10 presents the average of tracking errors over all people versus the total number of significant coefficients used in communication for our sparse representation framework. Since the total number of significant coefficients sent by a camera may change depending on the number of people at that moment, the maximum is shown in Figure 5.10. The performance of the distributed approach in [9] is also presented in Figure 5.10. It can be clearly seen that the distributed approach enables to make a huge reduction in communication in the network, but it cannot perform robust tracking. Our sparse representation framework achieves a bandwidth reduction less than the distributed approach, but, by using the custom-designed dictionaries, our framework has the ability to decrease the commu-

nication without affecting the tracking performance significantly. By using at least 50 coefficients per person, our sparse representation framework achieves an error of 2.5 pixel in the grid in average, whereas the distributed approach has a tracking error of 24 pixel in the grid in average. Our method is also advantageous over an ordinary decentralized approach that directly sends likelihood functions to the fusion node. In such an approach, we send each data point in the likelihood function, resulting a need of sending 6400 values for tracking four people. The performance of this approach is also given in Figure 5.10. Naturally, by using all the information in likelihoods, the ordinary decentralized approach achieves slightly better tracking performance than our method. However, our framework enables to decrease the communication in the network and achieve a tracking performance very close to performance of the original method.

The tracking results of the distributed approach in [9] per person and our sparse representation framework using 50 coefficients per person are given in Figure 5.11 and Figure 5.12, respectively. It can be seen that, the distributed approach fails to preserve identities. For all people in the scene, there occurs identity switches. One of the main reason of this problem is that the observations extracted from single view likelihood functions are not accurate and are not enough to represent the whole likelihood function. In the distributed approach, the fusion of observation is done after the observations are extracted from single view likelihoods. Since the observations are not extracted after fusing the multi-view likelihoods, noisy single view likelihoods creates inaccurate observations. An example for such a case is given in Figure 5.13. We can see that, since the observations extracted from likelihoods of each view are noisy, the estimated position of the person is not accurate. Hence, the

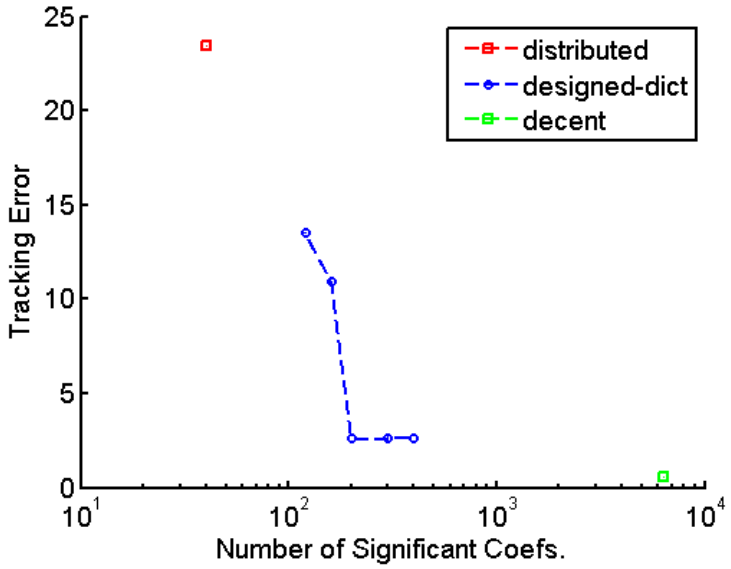
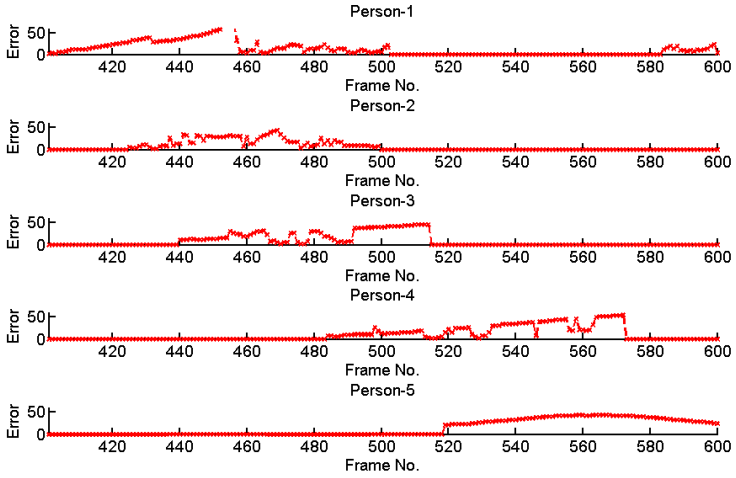


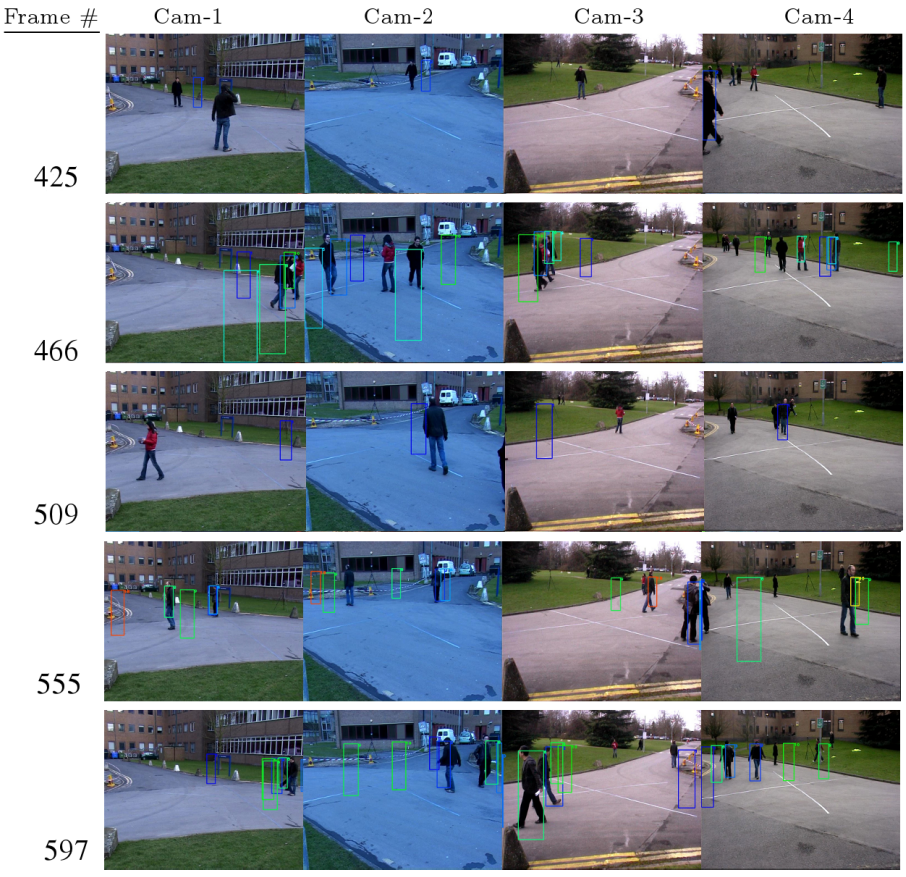

**Figure 5.10**: PETS 2009 sequence: The average tracking errors vs. the number of coefficients for the distributed approach in [9] (red), our sparse representation framework (blue) and a decentralized method (green) that directly sends likelihood functions.

distributed approach cannot robustly track the people. On the other hand, in our sparse representation framework, we first fuse the single view likelihoods and then use the multi-view likelihood function in tracking. By using the custom-designed dictionaries, we represent the likelihood functions with small number of coefficients without affecting the amount of information they contain. Thus, the multi-view likelihood obtained by fusing the reconstructed single view likelihoods is enough to perform robust tracking. In Figure 5.12, it can be seen that our framework can successfully preserve identities and track all people in the scene robustly. Even if a person leaves the scene and comes back (the first person in Figure 5.12-b), he or she is recognized and true label is assigned to the person. Because the tracking starts a few frames after the person enters the scene or ends before the person leaves the scene, we see some peak points in tracking error plots. When the number of coefficients taken per person are less then 50, we also observe identity problems. But by selecting the number of coefficients per person greater than or equal to 50, we can track all the people in the scene accurately.

In the light of the results we obtained, we can say that our sparse representation based method outperforms the distributed approach in [9]. By using the custom-designed dictionaries, we can both decrease the communication in the network and perform robust tracking. Our method requires only 3.12% of the bandwidth needed by the ordinary decentralized method in order to achieve a tracking performance very close to ordinary decentralized method. Another advantage of our framework is that it is possible to find a balance between robust tracking and reduction in communication. Whereas, in the distributed approach in [9] it is not easy to modify the algorithm in order to increase the tracking performance. The algorithm should be extended in

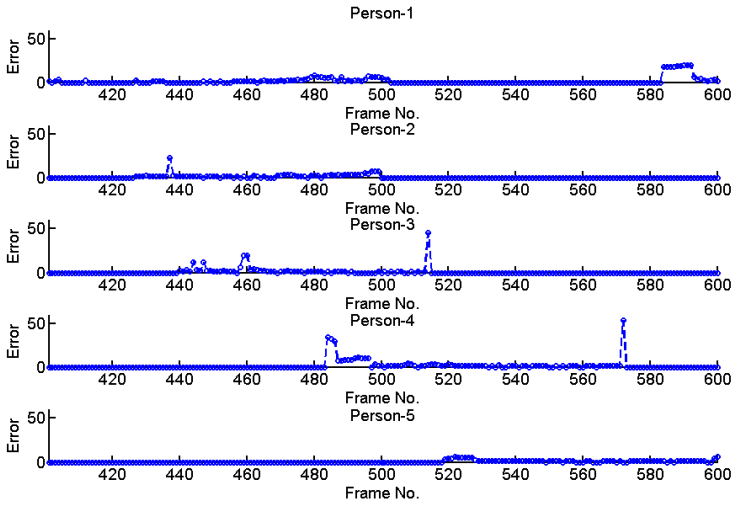



(a)




(b)

**Figure 5.11**: (a) The tracking errors for each person and (b) tracking results for the PETS 2009 dataset obtained by the distributed approach in [9].




(a)


(b)

**Figure 5.12**: (a) The tracking errors for each person and (b) tracking results for the PETS 2009 dataset obtained by our sparse representation framework using 50 coefficients per person used in communication.

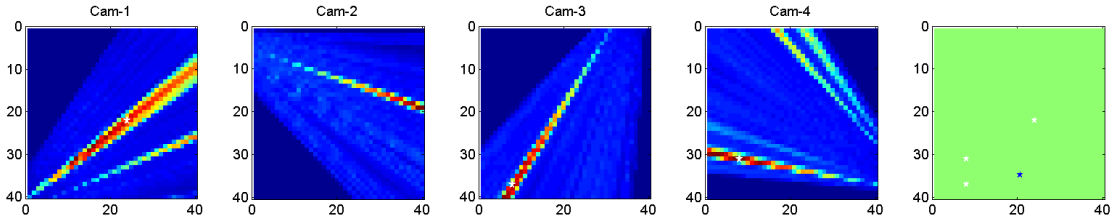



**Figure 5.13**: The illustration of the inaccurate observations in the distributed approach in [9]. White stars represent the observation extracted from likelihood function at each view and blue star represent the estimated position of the person.

a way to share more information between cameras, thereby obtain better tracking results.

# 6 CONCLUSION

## 6.1 Summary

Visual sensor networks (VSNs) constitute a new paradigm that merges two well-known topics: computer vision and sensor networks. A network of wirelessly connected cameras poses unique and challenging problems that do not exist either in computer vision or in sensor networks. VSNs consist of camera nodes, which integrate the image sensor, embedded processor, and wireless transceiver. Due to the resource constraints of the camera nodes, such as battery power and bandwidth, it is crucial to perform data processing and collaboration efficiently.

Over the last decade, an alternative sampling/sensing theory, known as compressed sensing, has been proposed to recover signals, images, and other data from what appear to be undersampled observations. Compressed sensing is a technique for acquiring and reconstructing a signal from small amount of measurements utilizing the prior knowledge that the signal has a sparse representation in a proper space. As a consequence, compressed sensing and sparse representation (SR) have become important signal recovery techniques because of their success for acquiring, representing, and compressing high-dimensional signals.

Following the success of sparse representation, in this thesis we have proposed three distinct sparse representation based ideas on existing problems of VSNs. The ability of sparse representation tools for reconstructing signals from small amount of observations fits well with the limitations in VSNs for processing, communication and collaboration. Hence, this thesis presents novel sparsity-driven methods that can be used to perform action recognition and human tracking applications in VSNs.

For action recognition, we have assumed that a test sample can be written as a linear combination of training samples from the class it belongs. In other words, a test sample has a sparse representation in the space covered by the training samples. Based on this assumption, we have cast the classification problem as an optimization problem and solved it by enforcing sparsity through $l_1$ regularization. We have developed two parallel perspectives one based on regular sparsity and the other one based on so called group sparsity. The sparse approach is based on the idea that a test sample can be represented by a small number of training samples, regardless of the class labels of the training samples. On the other hand, the group sparse approach imposes more structure, imposing sparsity across classes (i.e., allowing only a small number of classes to be active in the representation) while allowing the use of a large number of training samples from the active classes. Furthermore, a new approach has been proposed to adapt this method for VSN resource constraints. The role of sparsity in classification has been questioned in recent studies. It has been argued that the $l_1$-norm constraint may not be necessary. Following these studies, we have also analyzed the role of sparsity in classification for two different action recognition problems.

We have proposed a feature compression framework to overcome communication problems of human tracking systems in visual sensor networks. In our framework, we perform tracking in a decentralized way: each camera extracts useful features from the images it has observed and sends them to a fusion node which collects the multi-view image features and performs tracking. Usually, in tracking, extracting features results a likelihood function. Instead of sending the likelihood functions themselves to the fusion node, the likelihoods have been compressed by first splitting them into blocks, and then transforming each block to a proper domain and taking only the most significant coefficients in this representation. Sending the most significant coefficients to the fusion node has allowed us to decrease the communication in the network. To the best of our knowledge, compression of features computed in the context of tracking in a VSN has not been proposed in previous work.

As an extension of this framework, we have proposed a sparsity-driven approach for human tracking applications. Special overcomplete dictionaries that are matched to the structure of the likelihood functions have been designed and they have been used for sparse representation of likelihoods. In particular, we have designed our dictionaries to exploit the specific known geometry of the measurement scenario for the problem of human tracking. Each element in the dictionary for each camera corresponds to the likelihood that would result from a single human at a particular location in the scene. Hence, actual likelihoods extracted from real observations from scenes containing multiple individuals can be very sparsely represented in our approach. Thereby, we can decrease the communication between cameras and fusion nodes. To the best of our knowledge, this is the first method that uses sparse representation to compress likelihood functions and applies this idea for VSNs. This

framework fits well with the needs of the VSN environment. By extracting image features at the camera-level, the processing capabilities of cameras are utilized. Using only the most significant coefficients, obtained either from block-based compression scheme or sparse representation of likelihoods, in communication saves energy and bandwidth resources. We have achieved a goal-directed compression scheme for the tracking problem in VSNs by performing local processing at the nodes and compressing the resulting likelihood functions which are related to the tracking goal, rather than compressing raw images.

Another advantage of these frameworks is that they do not require the use of a specific tracking method. Since tracking methods usually process likelihood functions for estimation, our framework can work together with any kind of probabilistic tracking method. Existing methods can be used within our framework in VSN environments without making significant changes (e.g., using simpler features, etc.) which may degrade their performance. In our experiments, we have used two different tracking algorithms and achieved bandwidth reduction in the network without degrading the tracking performance significantly.

## 6.2 Future Directions

In Chapter 3, we have proposed a sparse representation based classification method for action recognition. But, we have also shown that rather than an $l_1$ constraint, it also is possible to obtain high level of accuracy using an $l_2$ constraint. Based on the results we have obtained for both 2-D and 3-D action recognition problems, saying one is better than the other is not possible. We believe it is essential to investigate

different classification problems in order to find special cases in which one is better than the other.

In chapter 5, we have designed overcomplete dictionaries for sparse representation of likelihood functions. As explained in Section 2.1.3, there are some approaches that are used to learn dictionaries from a training data in order to adapt the dictionaries to signals. It would be interesting to analyze the performance of dictionaries learned via such methods.

Furthermore, another approach for compressing likelihood functions in human tracking applications can be representing likelihoods as binary signals. Analyzing the performance of this approach and comparing with our feature compression and sparse representation frameworks would be a very intresting future direction.

In our sparse representation based likelihood compression framework, we design dictionaries according to the geometric configuration of cameras. Dictionary design is an off-line process that is only performed once before tracking starts. However, when the camera setup (e.g. camera location, camera view, etc.) changes, it is required to design the dictionaries again. It would be very interesting to work on a method to update the custom-design dictionaries according to the changes in camera setup.

In both action recognition and human tracking problems, the number of atoms of overcomplete dictionaries used in optimization problems (matrix $\psi$ in Eq. 3.8 and matrix $A_c$ in Eq. 5.1) are in the range of 10,000-50,000. In the case of much bigger dictionaries (e.g. matrices with more than 100,000 columns), standard well-known

algorithms (Section 2.1.2) may have difficulties in handling such very large-scale problems, thereby cannot work fast and efficiently. To overcome this issue, an interesting approach can be on exploiting the hierarchical structure of dictionaries [127, 128, 129, 130].

In chapter 4 and 5, we have proposed two frameworks for compressing likelihood functions in tracking applications in VSNs. In our frameworks, there is no assumption on the dimensionality of likelihood functions. We believe our frameworks can be used to compress any kind of likelihood function obtained by the measurements from any kind of sensor. Following this idea, another future direction would be to use our likelihood compression frameworks in wireless sensor network applications that includes different kinds of sensors, such as humidity sensors, temperature sensors, seismic sensors, etc.

# Bibliography


[1] M. Wakin, J. Laska, M. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. Kelly, and R. Baraniuk, "An architecture for compressive imaging," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 1273–1276.

[2] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83 –91, March 2008.

[3] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031 –1044, 2010.

[4] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," *Computer Vision Image Understanding*, vol. 104, pp. 249–257, November 2006. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2006.07.013

[5] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, 2004, pp. 32–36 Vol.3.

[6] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *PAMI*, vol. 30, no. 2, February 2008.

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

[8] "Eleventh ieee international workshop on performance evaluation of tracking and surveillance (pets)," http://pets2009.net, 2009.

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

[10] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[11] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 31, no. 2, pp. 210–227, Jun. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5454424

[12] P. V. Pahalawatta and A. K. Katsaggelos, "Optimal sensor selection for video-based target tracking in a wireless sensor network," in *in Proc. International Conference on Image Processing (ICIP ?04*, 2004, pp. 3073–3076.

[13] S. Fleck, F. Busch, and W. Straß er, "Adaptive probabilistic tracking embedded in smart cameras for distributed surveillance in a 3d model," *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, pp. 24–24, 2007.

[14] E. Oto, F. Lau, and H. Aghajan, "Color-based multiple agent tracking for wireless image sensor networks," in *ACIVS06*, 2006, pp. 299–310.

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

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

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

[18] L. Zhang, M. Yang, and X. Feng, "Sparse representation or collaborative representation: Which helps face recognition?" in *Computer Vision (ICCV), 2011 IEEE International Conference on*, nov. 2011, pp. 471 –478.

[19] Q. Shi, A. Eriksson, A. van den Hengel, and C. Shen, "Is face recognition really a compressive sensing problem?" in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, june 2011, pp. 553 –560.

[20] R. Baraniuk, E. Candes, R. Nowak, and M. Vetterli, "Compressive sampling [from the guest editors]," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 12 –13, march 2008.

[21] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397 –3415, dec 1993.

[22] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *Image Processing, IEEE Transactions on*, vol. 17, no. 1, pp. 53 –69, Jan. 2008.

[23] M. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995 –1005, June 2010.

[24] L. Potter, E. Ertin, J. Parker, and M. Cetin, "Sparsity and compressed sensing in radar imaging," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1006 –1020, June 2010.

[25] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," vol. 51, no. 1, pp. 34–81, 2009. [Online]. Available: http://dx.doi.org/doi/10.1137/060657704

[26] V. Cevher, A. C. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *ECCV*. Marseille, France: Springer, 2008, pp. 155–168. [Online]. Available: http://www.springerlink.com/index/l58k006877380905.pdf

[27] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, june 2008, pp. 1 –8.

[28] D. S. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards, and Practice*. Norwell, MA: Kluwer, 2001.

[29] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal $l_1$-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, 2006. [Online]. Available: http://dx.doi.org/10.1002/cpa.20132

[30] E. Candes and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Problems*, 2007.

[31] A. Yang, Z. Zhou, A. Balasubramanian, S. Sastry, and Y. Ma, "Fast $\ell_1$ -minimization algorithms for robust face recognition," *Image Processing, IEEE Transactions on*, vol. 22, no. 8, pp. 3234–3246, 2013.

[32] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004. [Online]. Available: http://www.worldcat.org/isbn/0521833787

[33] K. Frisch, "The logarithmic potential method of convex programming," University Institute of Economics (Oslo, Norway), Tech. Rep., 1955.

[34] N. Meggido, "Progress in mathematical programming interior-point and related methods," N. Megiddo, Ed. New York, NY, USA: Springer-Verlag New York, Inc., 1988, ch. Pathways to the optimal set in linear programming, pp. 131–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=72638.72646

[35] R. D. Monteiro and I. Adler, "Interior path following primal-dual algorithms. part i: Linear programming," *Math. Program.*, vol. 44, no. 1, pp. 27–41, Jun. 1989. [Online]. Available: http://dx.doi.org/10.1007/BF01587075

[36] M. Kojima, N. Megiddo, and S. Mizuno, "Theoretical convergence of large-step primal?dual interior point algorithms for linear programming," *Mathematical Programming*, vol. 59, no. 1-3, pp. 1–21, 1993. [Online]. Available: http://dx.doi.org/10.1007/BF01581234

[37] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[38] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale $l_1$-regularized least squares," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 606–617, 2007.

[39] M. Osborne, B. Presnell, and B. Turlach, "A new approach to variable selection in least squares problems," *IMA Journal of Numerical Analysis*, vol. 20, no. 3, pp. 389–403, 2000. [Online]. Available: http://imajna.oxfordjournals.org/content/20/3/389.abstract

[40] D. Malioutov, M. Cetin, and A. Willsky, "Homotopy continuation for sparse signal representation," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 5, 2005, pp. v/733–v/736 Vol. 5.

[41] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist*, 2004.

[42] M. D. Plumbley, "Recovery of sparse representations by polytope faces pursuit," in *in Proceedings of the 6th International Conference on Independent Component Analysis and Blind Source Separation (ICA 2006.* Springer Verlag, 2006, pp. 206–213.

[43] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004. [Online]. Available: http://dx.doi.org/10.1002/cpa.20042

[44] E. T. Hale, W. Yin, and Y. Zhang., "A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing," Rice University, Houston,TX, Tech. Rep., 2007.

[45] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, Jan. 2001. [Online]. Available: http://dx.doi.org/10.1137/S003614450037906X

[46] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1137/080716542

[47] S. Wright, R. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2479–2493, 2009.

[48] D. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, 1995.

[49] M. A. T. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 586–597, 2007.

[50] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/sqr(k))," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983. [Online]. Available: http://www.core.ucl.ac.be/~{}nesterov/Research/Papers/DAN83.pdf

[51] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2003.

[52] J. Yang and Y. Zhang, "Alternating direction algorithms for $l_1$-problems in compressive sensing," TR09-37, CAAM, Rice University, Tech. Rep., 2009.

[53] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045 –1057, 2010.

[54] Y. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, 1993, pp. 40–44 vol.1.

[55] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311 – 3325, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0042698997001697

[56] K. Engan, S. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 5, 1999, pp. 2443–2446 vol.5.

[57] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decomposition," in *Acous-*

*tics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 5, 2005, pp. v/293–v/296 Vol. 5.

[58] S. Sardy, A. G. Bruce, and P. Tseng, "Block Coordinate Relaxation Methods for Nonparametric Wavelet Denoising," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 361–379, 2000. [Online]. Available: http://www.jstor.org/stable/1390659

[59] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (gpca)," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 12, pp. 1945–1959, 2005.

[60] M. Aharon, M. Elad, and A. Bruckstein, "k -svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311 –4322, nov. 2006.

[61] E. Candes and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, 2008.

[62] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, and Y. Ma, "Towards a practical face recognition system: Robust registration and illumination by sparse representation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, pp. 597–604.

[63] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 643–660, 2001.

[64] D. L. Donoho and M. Elad, "Optimally sparse representation in general (non-orthogonal) dictionaries via $l_1$ minimization," in *Proc. Natl Acad. Sci. USA 100 2197?202*, Mar. 2003.

[65] E. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, Aug 2006.

[66] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr 2006.

[67] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, 1991, pp. 586–591.

[68] J. Kim, J. Choi, J. Yi, and M. Turk, "Effective representation using ica for face recognition robust to local distortion and partial occlusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 12, pp. 1977–1981, 2005.

[69] S. Li, X. Hou, H. Zhang, and Q. Cheng, "Learning spatially localized, parts-based representation," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–207–I–212 vol.1.

[70] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 684–698, 2005.

[71] O. Bryt and M. Elad, "Compression of facial images using the k-svd algorithm," *J. Vis. Comun. Image Represent.*, vol. 19, no. 4, pp. 270–282, May 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jvcir.2008.03.001

[72] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[73] J. Mairal, F. Bach, J. A. Ponce, G. Sapiro, and A. Zisserman, "Sdl: Supervised dictionary learning," in *in Advances in Neural Information Processing Systems, vol. 21,*, 2008.

[74] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling & Simulation*, vol. 7, no. 1, pp. 214–241, 2008. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/070697653

[75] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, pp. 1–22, 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/640386

[76] M. Taj and A. Cavallaro, "Distributed and decentralized multicamera tracking," *Signal Processing Magazine, IEEE*, vol. 28, no. 3, pp. 46 –58, may 2011.

[77] S. M. Khan and M. Shah, "Tracking multiple occluding people by localizing on multiple scene planes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 3, pp. 505–19, Mar. 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19147878

[78] M. Coates, "Distributed particle filters for sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*,

ser. IPSN '04. New York, NY, USA: ACM, 2004, pp. 99–107. [Online]. Available: http://doi.acm.org/10.1145/984622.984637

[79] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 181–188.

[80] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan, "Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons," in *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, 2007, pp. 219–226.

[81] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, dec. 2005, pp. 8179 – 8184.

[82] X. Wang, S. Wang, D.-W. Bi, and J.-J. Ma, "Distributed peer-to-peer target tracking in wireless sensor networks," *Sensors*, vol. 7, no. 6, pp. 1001–1027, 2007. [Online]. Available: http://www.mdpi.com/1424-8220/7/6/1001

[83] D. Yang, H. Gonzalez-Banos, and L. Guibas, "Counting people in crowds with a real-time network of simple image sensors," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct. 2003, pp. 122 –129 vol.1.

[84] D. Yang, J. Shin, A. O. Ercan, and L. Guibas, "Sensor tasking for occupancy reasoning in a camera network," in *Proceedings of IEEE/ICST 1st Workshop on Broadband Advanced Sensor Networks (BASENETS 2004)*, 2004.

[85] O. Javed, Z. Rasheed, K. Shafique, and M. Shah, "Tracking across multiple cameras with disjoint views," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 952–957 vol.2.

[86] H. Medeiros, J. Park, and A. Kak, "A light-weight event-driven protocol for sensor clustering in wireless camera networks," in *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, 2007, pp. 203–210.

[87] D. Lymberopoulos, T. Teixeira, and A. Savvides, "Macroscopic human behavior interpretation using distributed imager and other sensors," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1657 –1677, oct. 2008.

[88] T. Teixeira and A. Savvides, "Lightweight people counting and localizing in indoor spaces using camera sensor nodes," in *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, 2007, pp. 36–43.

[89] G. Srivastava, H. Iwaki, J. Park, and A. C. Kak, "Distributed and lightweight multi-camera human activity classification," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, aug. 2009, pp. 1 –8.

[90] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, 2005, pp. 65–72.

[91] C. Wu and H. Aghajan, "Real-time human pose estimation: A case study in algorithm design for smart camera networks," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1715 –1732, oct. 2008.

[92] X. Wang and S. Wang, "Collaborative signal processing for target tracking in distributed wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 67, no. 5, pp. 501–515, May 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2007.02.001

[93] I. Haritaoglu, D. Harwood, and L. Davis, "W4: real-time surveillance of people and their activities," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 809–830, 2000.

[94] S. Lu, J. Zhang, and D. Feng, "Classification of moving humans using eigen-features and support vector machines," in *Computer Analysis of Images and Patterns*, ser. Lecture Notes in Computer Science, A. Gagalowicz and W. Philips, Eds. Springer Berlin Heidelberg, 2005, vol. 3691, pp. 522–529. [Online]. Available: http://dx.doi.org/10.1007/11556121_64

[95] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *Decision and Control, 2007 46th IEEE Conference on*, 2007, pp. 5492–5498.

[96] M. Casares, S. Velipasalar, and A. Pinto, "Light-weight salient foreground detection for embedded smart cameras," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1223 – 1237, 2010, ¡ce:title¿Special issue on Embedded Vision¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314210001001

[97] S. Velipasalar, J. Schlessman, C.-Y. Chen, W. Wolf, and J. Singh, "A scalable clustered camera system for multiple object tracking," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, p. 542808, 2008. [Online]. Available: http://jivp.eurasipjournals.com/content/2008/1/542808

[98] S. Coşar and M. Çetin, "A group sparsity-driven approach to 3-d action recognition," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 1904–1911.

[99] A. Y. Yang, R. Jafari, S. S. Sastry, and R. Bajcsy, "Distributed Recognition of Human Actions Using Wearable Motion Sensor Networks," *Journal of Ambient Intelligence and Smart Environments*, pp. 1–5, 2009.

[100] H. Murase and S. Nayar, "Visual learning and recognition of 3-d objects from appearance," *International Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF01421486

[101] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 681–685, 2001.

[102] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[103] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," Stanford University, Tech. Rep., 2010.

[104] A. Majumdar and R. Ward, "Classification via group sparsity promoting regularization," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, April 2009, pp. 861–864.

[105] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 23, no. 3, pp. 257–267, Mar 2001.

[106] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep. 2013.

[107] M. S. Asif and J. Romberg, "Fast and accurate algorithms for re-weighted l1 norm minimization," *Submitted to IEEE Transactions on Signal Processing, July 2012*.

[108] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1137/080716542

[109] A. Yang, S. Sastry, A. Ganesh, and Y. Ma, "Fast $l_1$-minimization algorithms and an application in robust face recognition: A review," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 1849 –1852.

[110] J. Yao and J.-M. Odobez, "Multi-camera multi-person 3d space tracking with mcmc in surveillance scenarios," in *ECCV workshop on Multi Camera and Multi-modal Sensor Fusion Algorithms and Applications*, Oct. 2008.

[111] A. Gupta, A. Mittal, and L. Davis, "Constraint integration for efficient multiview pose estimation with self-occlusions," *PAMI*, vol. 30, no. 3, March 2008.

[112] M. Hofmann and D. Gavrila, "Multi-view 3d human pose estimation combining single-frame recovery, temporal integration and model adaptation," in *CVPR*, June 2009.

[113] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564 – 577, may 2003.

[114] T.-L. Liu and H.-T. Chen, "Real-time tracking using trust-region methods," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 3, pp. 397 –402, march 2004.

[115] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *COMPUTER VISON - ECCV 2002, PT 1*, ser. LECTURE NOTES IN COMPUTER SCIENCE, Heyden, A and Sparr, G and Nielsen, M and Johansen, P, Ed., vol. 2350. IT Univ Copenhagen; Univ Copenhagen; Lund Univ, 2002, pp. 661–675, 7th European Conference on Computer Vision (ECCV 2002), COPENHAGEN, DENMARK, MAY 28-31, 2002.

[116] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," in *Proceedings of the 6th international conference on Information processing in sensor networks*, ser. IPSN '07. New York, NY, USA: ACM, 2007, pp. 360–369. [Online]. Available: http://doi.acm.org/10.1145/1236360.1236406

[117] W. Wolf, B. Ozer, and T. Lv, "Smart cameras as embedded systems," *Computer*, vol. 35, no. 9, pp. 48–53, 2002.

[118] B. Tavli, K. Bicakci, R. Zilan, and J. Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools and Applications*, vol. 60, no. 3, pp. 689–726, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11042-011-0840-z

[119] I. Akyildiz, T. Melodia, and K. Chowdury, "Wireless multimedia sensor networks: A survey," *Wireless Communications, IEEE*, vol. 14, no. 6, pp. 32–39, 2007.

[120] G. Wallace, "The jpeg still picture compression standard," *Consumer Electronics, IEEE Transactions on*, vol. 38, no. 1, pp. xviii –xxxiv, feb 1992.

[121] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *Image Processing, IEEE Transactions on*, vol. 1, no. 2, pp. 205 –220, Apr. 1992.

[122] L. Winger and A. Venetsanopoulos, "Biorthogonal modified coiflet filters for image compression," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 5, May. 1998, pp. 2681 –2684 vol.5.

[123] S. Mallat, *A Wavelet Tour of Signal Processing, Second Edition (Wavelet Analysis & Its Applications)*, 2nd ed. Academic Press, Sep. 1999. [Online]. Available: http://www.worldcat.org/isbn/012466606X

[124] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, 1988.

[125] ——, *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611970104

[126] J. Yao and J. Odobez, "Fast human detection from videos using covariance features," 2008.

[127] K. Varshney, M. Cetin, J. Fisher, and A. Willsky, "Sparse representation in structured dictionaries with application to synthetic aperture radar," *Signal Processing, IEEE Transactions on*, vol. 56, no. 8, pp. 3548–3561, 2008.

[128] P. Jost, P. Vandergheynst, and P. Frossard, "Tree-based pursuit: Algorithm and properties," *Signal Processing, IEEE Transactions on*, vol. 54, no. 12, pp. 4685–4697, 2006.

[129] C. La and M. N. Do, "Signal reconstruction using sparse tree representation," in *in Proc. Wavelets XI at SPIE Optics and Photonics*, 2005.

[130] A. Shoa and S. Shirani, "Tree structure search for matching pursuit," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3, 2005, pp. III–908–11.