Particle filtering on large dimensional state spaces and applications in computer vision

by

Samarjit Das

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:
Namrata Vaswani, Major Professor
Yan Bin Jia
Nicola Elia
Zhengdao Wang
Aleksandar Dogandzic
Dan Nordman

Iowa State University

Ames, Iowa

2010

DEDICATION

This thesis is dedicated to Maa and Babu

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like express my gratitude towards my doctoral advisor, Prof. Namrata Vaswani for her guidance and mentoring over the years. Without her help and encouragement, this work would have never been completed. Also, I would like to thank Prof. Nicola Elia, Prof. Yan Bin Jia, Prof. Zhengdao Wang, Prof. Aleksandar Dogandzic and Prof. Dan Nordman for their suggestions and comments on my research. My special thanks to Dr. Shantanu Rane, Dr. Anthony Vetro and Mitsubishi Electric Research Laboratories (MERL) at Cambridge, MA for a great research internship. Thanks also to our research collaborator, Dr. Amit Kale of Siemens Corporate Technologies, Bangalore.

I thank my friends and fellow graduate students for their support and camaraderie over the years. I, especially, thank Neevan and Naveen (Goli) for being great friends of mine for all these years; I would also like to thank my friends - Ranojoy, Amit, Vamsi, Fardad, Sambarta, Syaan, Ashish (Lion) and Hemant (Pirate) - all of them made my ISU and Ames experience very memorable. Thanks also to Shubhro, my roomie during the entire stay at Ames for all his support and friendship. My present and former colleagues in the CSP group also deserve a special mention. My thanks to Wei Lu, Chenlu Qui, Burcu and Toran Li for their support and feedback on my work.

Words are not enough to express my gratitude towards my family. My sincerest appreciation to my parents, Mr. Monoranjan Das and Mrs. Simu Das, for instilling the basic virtues that I rely on so heavily and being a constant source of support and encouragement. They have been the greatest inspiration for me in every aspect of life. My sisters, Mallika (didibhai) and Tanushree (Choordibhai) have always been there when I need them. They have had a very positive influence on my life and I can't thank them enough for that. It is following their

footsteps that I started my journey into academia. Wherever I have reached and whatever I have achieved so far, it is all because of the caring support from my family. They are the driving force behind my journey which I started in a small town called Hojai in the state of Assam (India).

Thanks a lot Iowa State University! Finally, thank God for the life, all the opportunities and a wonderful family. I couldn't have asked for more.

ABSTRACT

Tracking of spatio-temporal events is a fundamental problem in computer vision and signal processing in general. For example, keeping track of motion activities from video sequences for abnormality detection or spotting neuronal activity patterns inside the brain from fMRI data. To that end, our research has two main aspects with equal emphasis - first, development of efficient Bayesian filtering frameworks for solving real-world tracking problems and second, understanding the temporal evolution dynamics of physical systems/phenomenon and build statistical models for them. These models facilitate prior information to the trackers as well as lead to intelligent signal processing for computer vision and image understanding.

The first part of the dissertation deals with the key signal processing aspects of tracking and the challenges involved. In simple terms, *tracking* basically is the problem of estimating the hidden state of a system from noisy observed data(from sensors). As frequently encountered in real-life, due to the non-linear and non-Gaussian nature of the state spaces involved, Particle Filters (PF) give an approximate Bayesian inference under such problem setup. However, quite often we are faced with large dimensional state spaces together with multimodal observation likelihood due to occlusion and clutter. This makes the existing particle filters very inefficient for practical purposes. In order to tackle these issues, we have developed and implemented efficient particle filters on large dimensional state spaces with applications to various visual tracking problems in computer vision.

In the second part of the dissertation, we develop dynamical models for motion activities inspired by human visual cognitive ability of characterizing temporal evolution pattern of shapes. We take a landmark shape based approach for the representation and tracking of motion activities. Basically, we have developed statistical models for the shape change of a

configuration of "landmark" points (key points of interest) over time and to use these models for automatic landmark extraction and tracking, filtering and change detection from video sequences. In this regard, we demonstrate superior performance of our Non-Stationary Shape Activity(NSSA) model in comparison to other existing works. Also, owing to the large dimensional state space of this problem, we have utilized efficient particle filters(PF) for motion activity tracking. In the third part of the dissertation, we develop a visual tracking algorithm that is able to track in presence of illumination variations in the scene. In order to do that we build and learn a dynamical model for 2D illumination patterns based on Legendre basis functions. Under our problem formulation, we pose the visual tracking task as a large dimensional tracking problem in a joint *motion-illumination* space and thus use an efficient PF algorithm called PF-MT(PF with Mode Tracker) for tracking. In addition, we also demonstrate the use of change/abnormality detection framework for tracking across drastic illumination changes. Experiments with real-life video sequences demonstrate the usefulness of the algorithm while many other existing approaches fail. The last part of the dissertation explores the upcoming field of compressive sensing and looks into the possibilities of leveraging from particle filtering ideas to do better sequential reconstruction (i.e. tracking) of sparse signals from a small number of random linear measurements. Our preliminary results show several promising aspects to such an approach and it is an interesting direction of future research with many potential computer vision applications.

CHAPTER 1. Introduction

One key aspect to human intelligence is our ability to interpret the visual world around us. This, in turn, comes from our ability to decode meaningful information embedded in the spatio-temporal variations of signals e.g. sequence of images formed in our retina. Our world is scattered with signals like these - from surveillance videos to medical imaging or even signals from the outer space. These signals are important because their evolution patterns capture vital information about the underlying physical processes. One of the primary goals of computer vision is to mimic our brain's remarkable ability to reveal rich information about the physical processes from spatio-temporal data or in other words, assigning 'meaning' to the visual world.

From a signal processing standpoint, the physical processes, for example, could be the motion activity performed by an individual being observed by a video camera or it could be the neuronal activities inside the brain being studied by Functional Magnetic Resonance Imaging (fMRI). And, our goal could be to *keep track* of motion activity from the video or *follow* neuronal activity patterns from fMRI data. Now, this is where 'tracking' comes into picture. Tracking is a very useful statistical signal processing technique being used by the computer vision community over the last couple of decades. Physically, tracking means: following the state of an entity from noisy observed data. The state is essentially hidden from the observer. In the context of above discussion, the state is what represents the underlying physical processes. For example, the neuronal activity pattern is 'somehow' related to the fMRI data but not directly accessible from it or, the motion activity of various body parts is not 'directly obvious' from the temporal intensity patterns of a set of pixels in the video - and hence, we need to estimate the 'hidden' state from the noisy observation data if we want to extract

meaningful information about the physical system i.e. learning the brain's response to external stimuli or motion activity recognition/tracking from the videos. Now, when we talk about the relationship between the hidden state and the noisy observed data, there are two aspects to it. First, the observation process or observation extraction mechanism. Depending on various applications, it can vary a lot. For example, the particular techniques used for observation extraction in fMRI leads to a specific relationship between neuronal activity and the observed fMRI data. Similarly, there is a completely different mapping that maps a specific motion activity to a time varying pixel intensity patterns. Now, most often, we have an inadequate knowledge about this relationship and only use a model, known as the observation model, in order to fit the observation process as closely as possible. This modeling error essentially is one aspect of the noise associated with the observation data - here, noise means the unknown aspects of the observation process. Second, is the noise incorporated in the physical process of observation data acquisition. It could be anything from sensor noise in the video camera and thermal noises associated with fMRI instrumentation to inevitable physical interferences like occlusion and clutter. In general, an ideal observation model would attempt to incorporate all these issues.

Thus, tracking is the problem of causally estimating a hidden state sequence from a sequence of observations. The role of the observation model is extremely crucial in this problem. However, under a Bayesian framework, the trackers also use the prior knowledge on the state process often called the system model. This guide the tracker towards a more informative decision about the hidden state. The corresponding class of tracking algorithms is popularly known in the signal processing community as Bayesian Filtering. Some examples are - Kalman Filter(KF) [1], Extended Kalman Filter(EKF) [1], Particle Filter(PF) [2, 3] etc. Now, the major bottleneck of using KF and EKF is that, quite often, the observation extraction associated with natural processes leads to highly non-linear observation model. Together with this the observation noise also turns out to be non-Gaussian because the interferences like occlusion and clutter. Also, sometimes, the system model or state dynamical model associated with the problem tends to be nonlinear - e.g. body dynamics for various motion activities. It has been

shown in recent years that under a non-linear, non-Gaussian problem setup, Particle Filter best suits the job of a tracker. But again, it has its own limitations when it comes to dealing with large dimensional state processes and tackling the effects of occlusion and clutter. This leads to a major bottle neck to the use of PFs as generic tracking framework.

Particle Filtering (PF) has been very popular in the computer vision community in recent years. It has been used from visual tracking for surveillance applications to mobile robot localization and path planning. But still, traditional PF becomes very inefficient while dealing with large dimensional problems (generally, state space dimension of 10 or more). Large dimensional state estimation is not just a theoretical problem in signal processing; rather, it is very frequently encountered in real-life tracking applications. For example, the tracking motion activities of the human body (more than 10 body joints to keep track of) or tracking complex illumination patterns of a scene (large dimensional illumination parameter vector to be tracked).

The reason behind this large dimensionality issues with PFs traces back to the fact that the PF essentially is a sequential Monte Carlo technique. And, as the state space dimension increases, the required number of particles for maintaining tracking accuracy drastically increases (i.e. the effective particle size decreases). Thus, the performance of the traditional PFs takes a severe beating under limited particle budget (under memory/processing speed constraints) for large dimensional state space problems. Together with this, the interferences of occlusion and clutter is inevitable in the observation process. From a signal processing standpoint, the ramification of all this: a multimodal observation likelihood. Both, large dimensional state space and multimodal observation likelihood poses significant signal processing challenges which are out of the scopes of traditional PFs [2, 3].

This brings us to the main focus of our research and *organization* of the thesis. The dissertation has four parts. In the first part of the dissertation we discuss how we have developed efficient PF algorithms which can work on large dimensional state space(LDSS) as well as handle multimodal observation likelihood due to occlusion and clutter. To that end, we have used these efficient PF algorithms for various real-life computer vision problems like - motion

activity tracking and automatic landmark extraction from videos and visual tracking under illumination variations etc. This leads us to the rest of the dissertation. The second part of the dissertation focuses on our extensive research on development of dynamical models for motion activities based on statistical landmark shape analysis. These models serve as the state transition prior(STP) in the efficient particle filtering framework for tracking motion activities from videos, abnormality detection and in many other computer vision applications. In the third part, we tackle the problem of visual tracking under illumination variations. This is a challenging problem because, we have to perform tracking in a joint motion-illumination space leading to a large dimensional state space. But again, this is exactly where our efficient PF algorithms can do a great job. Another contribution of this work is the development of a dynamical model for illumination pattern changes in a scene using Legendre basis functions and the application of change detection framework for tracking across drastic illumination changes. The fourth and the final part of the dissertation is exploratory, open ended and we believe, is a promising direction for future work. Based on the upcoming field of compressive sensing, we take a new approach towards recursive reconstruction (i.e. tracking) of 'large dimensional' sparse signal sequences. We discuss more details about this in the sections to follow.

1.1 Motivation

In this section, we discuss the problems more in details and explain what motivated us towards solving them. First, we discuss major issues/challenges behind developing efficient PFs for large dimensional state spaces. Then we carefully look into the intuitions and practical considerations that led us to the development of dynamical models for motion activities as well as modeling illumination dynamics in the scene. We also look into what all practical applications they might be useful in. Finally, we give a brief outline of our work on particle filtered compressive sensing and motivate why and where it might be useful.

1.1.1 Why Need Efficient PFs: The LDSS Problem

As we discussed earlier, tracking is the problem of causally estimating a hidden state sequence from a sequence of observations that satisfy the Hidden Markov Model (HMM) assumption. A tracking algorithm recursively computes the "posterior" at time t (probability density function of the current state conditioned on all observations until t) using the posterior at the previous time and the current observation. For linear and Gaussian state space models, the posterior is Gaussian and the Kalman filter provides closed form sequential solutions for computing its parameters (mean, covariance) at each t. But, for most nonlinear or non-Gaussian state space models, the posterior cannot be computed analytically. But, it can be efficiently approximated using the particle filter (PF) [2, 4] which is a sequential Monte Carlo technique. A PF outputs at each time t, a cloud of N "particles" (Monte Carlo samples), along with their corresponding weights, whose empirical measure closely approximates the true posterior for large N.

An important issue in PF design is to choose an importance sampling density that reduces the variance of the particle weights and thus improves "effective particle size" [3]. The original PF [2] used the state transition prior as the importance density which assumes nothing and is easiest to implement. But since it does not use knowledge of the current observation, the weight variance can be large (particularly when the observations are more reliable compared to the prior model), resulting in lower effective particle sizes [4]. The "optimal" importance density [3] is the posterior conditioned on the previous state (denote it by p^*). When p^* is unimodal, PF-Doucet [3] approximates it by a Gaussian about its mode (Laplace's approximation [5]) and importance samples from the Gaussian. However, for most practical applications, p^* turns out to be multimodal. We discuss this more in detail in the contribution section to follow. This is a major challenge for efficient PF implementation. Also, when in addition to multimodality, the state space space dimension is large (typically more than 10 or 12), the effective particle size reduces, i.e. the number of particles required for reasonable accuracy becomes very large [2, 6]. This makes a regular PF impractical. Owing two these two main challenges in PF literature, we have developed efficient PF algorithms that tackle them. Our contributions in

this area are discussed in the section to follow.

1.1.1.1 Efficient PFs : Applications of PF-EIS, PF-MT, PF-EIS-MT

As mentioned earlier, when the optimal importance density (denoted as p^*) of the PF is unimodal, it can be approximated as a Gaussian and generate importance samples from it as done in [3]. Other work in PF literature that also implicitly assumes that p^* is unimodal includes [4, 19, 20]. But very often, the observation likelihood (OL) is multimodal or heavy tailed, e.g. due to clutter, occlusions or low contrast/blur in image sequences. If the state transition prior (STP) is broad compared to the distance between OL modes, it is easy to see that p^* will be multimodal [21] (see Fig. 2.2). For such problems, we have proposed [22, 23, 21, 15] Particle Filter with Efficient Importance Sampling(PF-EIS) algorithm in which combines PF-Original [2] with PF-Doucet [3]. We also demonstrate the applications of PF-EIS while dealing with multimodal observation likelihood for tracking landmark shapes in video sequences [22, 24]

In order to tackle the large dimensionality of the state space, several approaches have been used in recent years. For example, one solution that partially addresses this issue is [25, Ch 13] or [26] which propose to resample more than once within a time interval. But more resampling results in more particle degeneracy [4]. In the special case when the state space model is conditionally linear-Gaussian, or if these states can be vector quantized into a few discrete centers, Rao Blackwellization (RB-PF) [27, 6] can be used. In general, neither assumption may hold. But in most large dimensional state space(LDSS) problems, state change variance is large in only a few dimensions while in the rest, the state change is quite small (LDSS property) [21]. We exploited this property in [21] to introduce a mode tracking (MT) approximation of importance sampling (IS), which greatly reduced the IS dimension. In other words, if the STP is narrow enough, then there will be small error in replacing importance sampling (IS) by mode tracking (MT) in these directions. In MT, we deterministically set the residual state particle equal to the the mode of the residual posterior. The MT step thus significantly reduces the importance sampling dimension of the resulting PF, consequently improving its effective

particle size. As an extension to this, we also combine the EIS and MT ideas and develop the PF-EIS-MT algorithm which is expected to be more efficient than PF-EIS and PF-MT under certain conditions. To sum up, our main contribution has been to take these efficient PF algorithms and implement them for solving real-world computer vision problems like motion activity tracking from videos and visual tracking under illumination variations. These problem pose the challenges of large dimensional state space as well as multimodal observation likelihood making the use of efficient PFs very crucial for solving them.

1.1.2 Dynamical Model for Motion Activities and Applications

First, let us give the main motivations behind this work and the challenges involved. Here's why we would like to build dynamical models for motion activities. The fundamental idea behind this research is actually motivated by ability of human visual cognition. It is to be noted that as interesting as it is our ability to recognize objects without any difficulty, our brain also has a remarkable ability to recognize/classify temporal evolution patterns of things - e.g. shapes or may be collective evolution pattern of a group of points. Just by looking at video and seeing how body shapes evolve over time we can easily figure out about motion activity - running, jumping, dancing and so forth. It is also important that our interpretation about motion activities has an invariance aspect w.r.t position, scale and rotation; meaning - move, zoom or rotate the camera : does not effect the motion activity as we perceive it. Question is - How do our brains do that ? Here's an example. Say, someone provided us with the locations of various human body joints in a black background as someone performs an activity. Interestingly, it has been found that we can recognize (or, at least guess that it is a human body in action) the motion activity just from the temporal evolution patterns of those points. This has been know to the computer vision/graphics community for quite a while. We are just good at it, even without the actual video of the person! - we call these type of points as landmark points -key points of interest on the body e.g. various body joints corresponding to hand, knee, shoulder etc.

In general, landmarks are the key feature points of interest in a video or image sequence.

As shown in Fig. 1.1, it could be the body joints for motion capture or it could be facial points for performance capture (as used in the movie AVATAR) or some times a single object like a car or a person could be treated as a landmark point and our goal could be to study the collective behavior of a group of interacting point objects over time. Nevertheless, it is evident that the temporal evolution of a group of landmark points has some useful *motion activity specific* information embedded in it and what is interesting is that our visual cognition captures it without any other visual references. And, we would like to build a dynamical model - a time series model for the shape change of a configuration of "landmark points in order to capture this information. Now, what do we do with these models? As we will see that these models can be very effective to be used as the prior models for motion activities in a visual tracking framework and more. To that end, there are two main challenges involved when it comes to the use of this dynamical model for motion activity analysis or tracking. First, the dynamical model for landmark shape change should have generic framework allowing large to arbitrary shape variations over the entire sequence, as expected in real-life motion activities. Second, in order to apply these models in a visual tracking framework our tracking algorithm has to a) efficiently work large dimensional state spaces(as the number of landmark points could be large) and b) able to handle the effects of occlusion and clutter (see the challenges involved in Fig. 1.2). We have used efficient particle filers developed in our earlier research for getting around these challenges.

In the context of the above discussion, the goal of this work is to develop statistical models for the shape change of a configuration of "landmark" points (key points of interest) over time and to use these models for filtering, tracking (to automatically extract landmarks), synthesis and change detection applications. "Shape" of an ordered set of landmarks was defined by Kendall [7] as all the geometric information that remains when location, scale and rotational effects are filtered out. The term "shape activity" was introduced in [8] to denote a particular stochastic model for the dynamics of "landmark shapes" (dynamics after global translation, scale and rotation effects are normalized for). A model for shape change is invariant to camera motion under the weak perspective model (also referred to as the scaled orthographic camera)

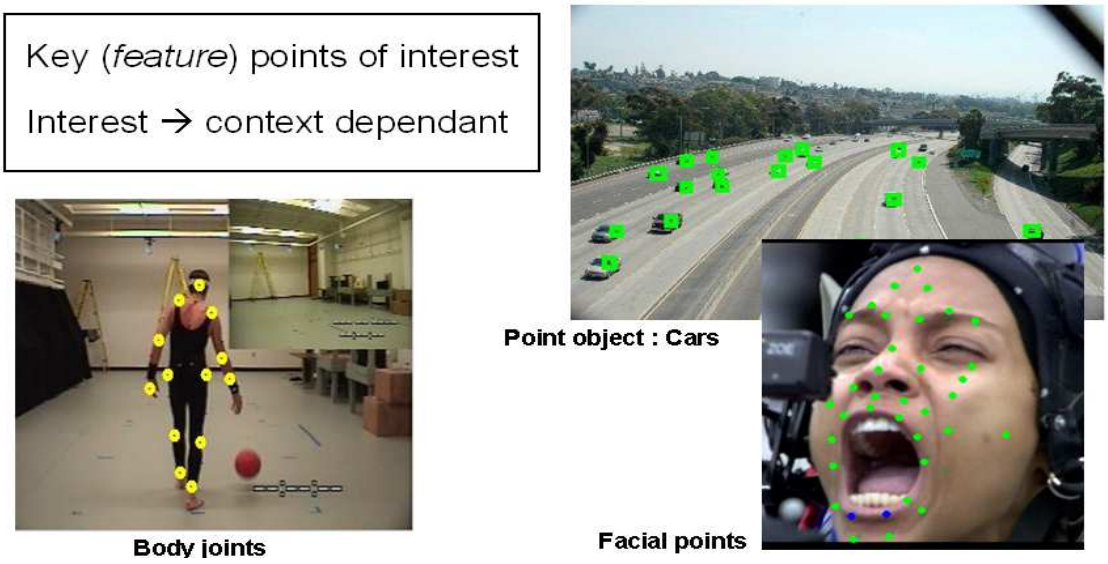


Figure 1.1 Landmark points. Basically, they are the key feature points of interest in a video or image sequence.

[9] which is a valid assumption when the scene depth is small compared to distance from the camera. The models studied in [8] were primarily for modeling stationary shape activities (SSA) of 2D landmarks (assume constant "mean shape"). In this work we propose models for the dynamics of nonstationary shape sequences (referred to as "nonstationary shape activities" (NSSA)) of 2D and of 3D landmarks. Most "activities" of a set of landmarks, for e.g. see Fig. 8, are not stationary and hence this more general model is needed. Even if the activity is actually stationary it still gets tracked using our model.

2D landmarks are usually the 2D coordinates of feature points of interest in an image sequence, for example these could be the joints of the different body parts of the human body and the goal could be to model and track articulated human body motion (see Fig. 3.21, Fig. 3.22). Alternatively, these could be the locations of a set of interacting point objects and the goal could be to track their collective behavior over time and detect abnormalities [8]. 3D landmark shape sequences are often obtained from a time sequence of volume images, for example by manually or automatically extract landmarks from a 3D heart MR image sequence or from a time sequence of brain MRI volumes. 2D or 3D landmarks may also be obtained from motion capture (MOCAP) [10] data where sensors are attached to various joints of the human

body and their 3D coordinates measured over time. The Carnegie Mellon Motion Capture database (CMU-MOCAP) is a common example. Modeling Mocap data has applications in bio-mechanics and graphics to understand the motion of human joints in various actions. Some of the applications are demonstrated in Fig. 1.3.

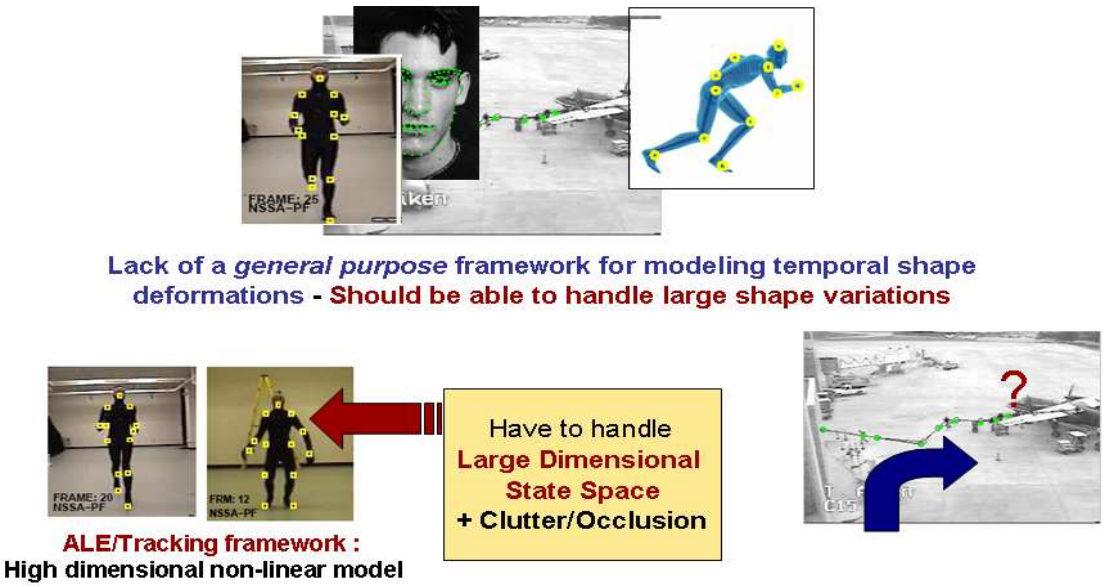


Figure 1.2 The challenges involved in the problem of tracking motion activities based on landmark shapes

A shape activity model serves as the prior for model-based automatic tracking or filtering of landmark shape sequences. We demonstrate this in chapter 3, Fig. 3.21, Fig. 3.22 for human activity videos taken from CMU-MOCAP database. Additionally, as demonstrated in [11], it can also be used for model-based compression of the shape sequence and this is useful in greatly reducing the amount of data to be stored or transmitted across a network. A third application is in graphics to synthesize new sequences for animation or to extrapolate existing sequences, for e.g. see Fig. 3.23. A fourth key application is in model based change detection or abnormality detection problems, e.g. [12, 8] and Fig. 3.24.

1.1.3 Visual Tracking Under Illumination Changes

Tracking illumination changes of moving objects is a challenging problem, particularly when the illumination is different in different regions of the object. Particularly, template matching

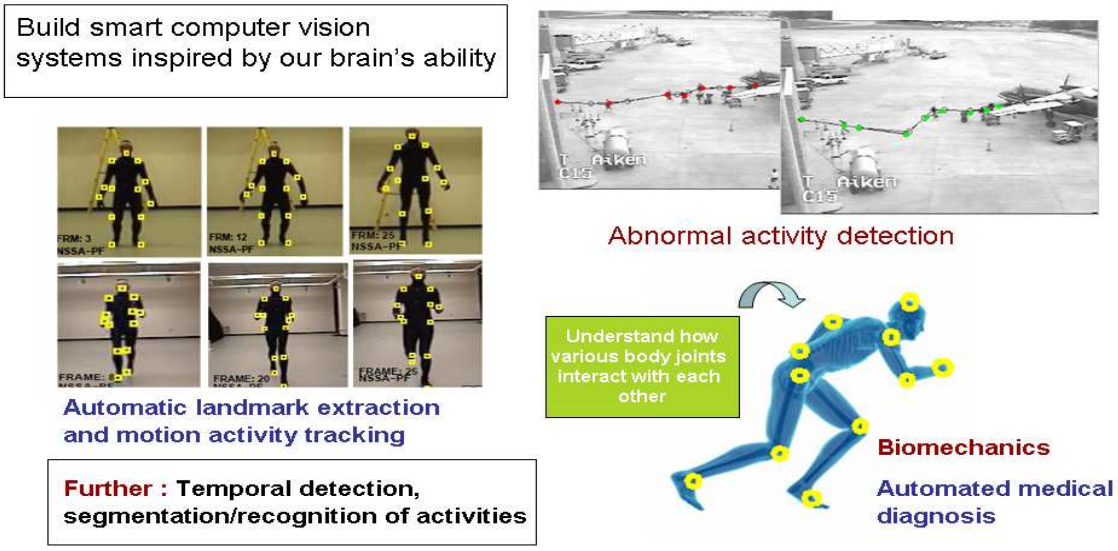


Figure 1.3 Various applications of modeling landmark shape dynamics. These include motion activity tracking from videos, automatic landmarks extraction, abnormality detection and even in biomechanics to study the interaction among various body parts.

based tracking frameworks fail due to the object appearance change because of illumination changes (shown in Fig. 1.4). This necessitates illumination tracking along with object motion. In the absence of illumination change, the motion of a rigid object moving in front of a camera that is far from the scene can be tracked using a three dimensional vector consisting of x-y translation and uniform scale or more generally using a six dimensional affine model as in Condensation [13]. Condensation beautifully demonstrated the use of a particle filter (PF) for tracking through multimodal observation likelihoods resulting from background clutter or occlusions. Now if illumination also changes over time and if different parts of the object experience different lighting conditions, then more dimensions get added to the state space (An example is shown in Fig. 1.5). Even a simple model of illumination such as that used in [14], which parameterizes illumination using a Legendre basis, requires a 3-7 dimensional basis to represent illumination accurately. But even a 7-dimensional basis will increase the total state space dimension to between 10 and 13.

It is well known that as the state space dimension increases, the number of particles required to track using a PF increases [3], thus making PF impractical for larger dimensional problems.

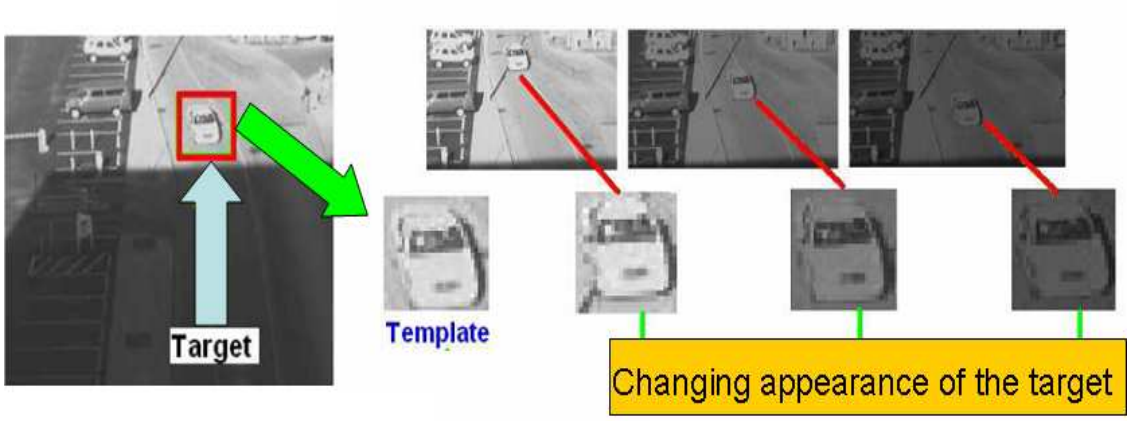


Figure 1.4 The failure of template matching based trackers due to illumination changes.

But, in most practical cases, the posterior of illumination change, conditioned on motion and previous state, is unimodal. When there is no occlusion, this is true because even though the likelihood may be multimodal as a function of the motion parameters, e.g. if there are more than one objects in the scene that match the template, it will be unimodal conditioned on them. Even when there is occlusion, as we explain later in chapter 4, the conditional posterior of illumination will still be unimodal, as long as the occluding object intensity is different enough from that of the object of interest. Also the illumination change over time is usually very gradual and this results in a very narrow posterior. Under these two assumptions, a recently introduced posterior mode tracking (MT) idea [15] can be adapted for tracking the illumination state. This one step, reduces the importance sampling dimension to 3 (or 6) instead of 10 (or 13), thus drastically reducing the number of particles required. We refer to the resulting PF as Illumination PF-MT.

In designing the Illumination PF-MT, we use a simple motion model and use a Legendre basis for parameterizing illumination change [14]. We assume a simple Gaussian random walk prior on both the motion and the illumination state vectors. The illumination change covariance learnt from training data is usually very small. The exception is when the object transitions from shadow to sunlight or vice versa or in an indoors scenario when a light bulb is switched off or on. During the transition phase, one needs a large illumination change

Figure 1.5 Illumination variations on the face. Observe that illumination conditions experienced by different parts of the face is different.

covariance to remain in track. This situation can be well modeled by a random walk model with two values of covariance - a large covariance (or in effect a weak prior) when "transition" is detected and a much smaller covariance (learnt from training data) when "no transition" is detected. One can detect the beginning and end of "transition" using a change detection statistic. We would like to use a statistic that detects change before significant loss of track occurs, so that can detect the change and to compensate for it without having to reinitialize the tracker. The recently proposed generalized ELL (gELL) statistic [16] does exactly this since it uses the tracked part of the change to detect it. We demonstrate successful tracking through light to shadow transitions using the gELL statistic to detect the transitions before loss of track. This is done without any need for reinitialization.

1.1.4 Particle Filtered Modified Compressive Sensing (PaFiMoCS)

The theory of compressive sensing(CS) says that a large dimensional sparse signal can be reconstructed from a set of highly undersampled random measurements by using convex optimization techniques. In recent literature, [17, 18] proposes that if we have a partial knowledge of the support of the signal, a modified version of compressive sensing (called mod-CS) can reconstruct the signal from even lesser number of measurements. This is particularly important for sequential reconstruction of sparse signals where, the support at the previous instant can be used as the predicted support and perform modCS. Obviously, this works great when we

have a slowly varying sparsity pattern i.e. there is a small number of addition and deletion to the support over successive instants. But what if this 'slow variation' assumption is not true ? - can we still do modCS by devising a mechanism to make a reasonable prediction about the support ? PaFiMoCS basically attempts to answer this question. Under the assumption that the sparsity pattern variation might evolve from a dynamical model, PaFiMoCS uses the idea of sequential importance sampling from particle filtering literature, and provides the modified compressive sensing with a close enough estimate of the support. This enables mod-CS style algorithms to work more efficiently i.e. using lesser number of observations. Together with this, a dynamic model on the signal value change on the known support also enables recently proposed regularized modified compressive sensing (reg-modCS) to be used under this framework, which tends to be more robust to the observation noises.

1.2 Thesis Outline

In this section, we give an outline of the thesis. In Chapter 2, we discuss the basics behind particle filtering and move on to the development of efficient particle filters for large dimensional state spaces. In Chapter 3, we develop statistical dynamical models for motion activities using landmark shape deformation models and use it for filtering, tracking to automatically extract landmarks, change detection and landmark shape data compression. In Chapter 4, we propose a visual tracking system that handles illumination variations in the scene. To that end, we develop a dynamical model for illumination changes and use efficient particle filter for tracking in a joint motion-illumination space. Together with this, we also demonstrate the use of change/abnormality detection for tracking across drastic illumination changes. In Chapter 5, we develop a novel algorithm for recursive reconstruction of a sparse signal sequence using modified compressive sensing facilitated by a particle filtering step for support prediction. Finally, we conclude the thesis in Chapter 6 and give several directions for future research.

CHAPTER 2. Efficient Particle Filtering

In this chapter, we first go through some of the basic concepts behind Particle Filter and in general, sequential Monte Carlo techniques for Bayesian Filtering(taken from [3]). We then move on to the signal processing aspects and challenges involved in traditional Particle Filters, particularly in the case of large dimensional state space and multimodal observation likelihood. Finally, we propose efficient particle filters to tackle these challenges.

2.1 Basics : The Bayesian Filtering Problem

Consider a state-space model with $t = 0, \ldots, T$,

$$X_t = \phi(X_{t-1}) + n_t \tag{2.1}$$

$$Y_t = g(X_t) + v_t \tag{2.2}$$

where X_t and Y_t denote the state and the observation at the current instant respectively [3]. In real-life applications, for example, the state can be the position of a target while the observation is the noisy sensor data about the current position and our goal could be to extract the true 'hidden' state information using the observations and the state dynamical model (the system model, eq (3.27)). The functions $\phi(.)$ and $g(.)$ can be any linear or non-linear function. The following things can be assumed to be known. The initial state distribution $p(X_0)$, the state transition density $p(X_t|X_{t-1})$ and the observation likelihood $p(Y_t|X_t)$. Here, $p(.)$ is used to denote probability density function. The transition density and the observation likelihood can be determined form the know distributions of i.i.d noise sequences n_t and v_t. The state space is assumed to follow a hidden Markov model (HMM). In other words, the state sequence X_t is a Markov process and the observations Y_t, $t = 1, 2, ...$ are conditionally independent

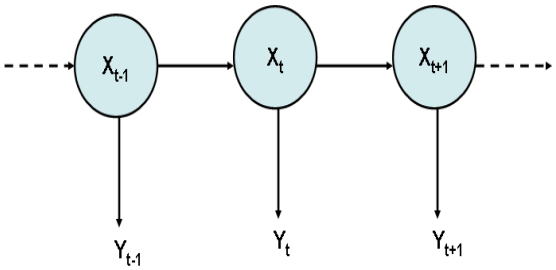


Figure 2.1 This figure shows the hidden Markov model structure of the state space. The state sequence X_t is a Markov process and the observations Y_t, $t = 1, 2, ...$ are conditionally independent given the state at t

given the state at t. The states are hidden (i.e. not observable); all we can know about the system is through the observations known at each instant. The graphical model of the HMM is shown in Fig. 2.1. Under HMM assumption, $p(X_t|X_{t-1}, past) = p(X_t|X_{t-1})$ and $p(Y_t|X_t, past) = p(Y_t|X_t)$ e.g. $p(Y_t|X_t, Y_{t-1}) = p(Y_t|X_t)$.

The goal of particle filtering (or bayesian filtering) is to estimate the joint posterior state distribution at time t i.e. $p(X_{1:t}|Y_{1:t})$ or quite often its marginal $p(X_t|Y_{1:t})$. Here, $X_{1:t} = \{X_1, X_2, ..., X_t\}$. Finally, we would like to compute

$$I_t = E_{p(X_t|Y_{1:t})}(f(X_t)) = \int f(X_t)p(X_t|Y_{1:t})dX_t \tag{2.3}$$

When the posterior can be assumed to be Gaussian and function $\phi(.)$ and $g(.)$ to be linear the same problem can be recursively solved using Kalman filter [1]. To tackle non-linearity of ϕ and $g(.)$, the Extended Kalman filter [1] can be used, but it still assumes the gaussianity of the posterior distribution. But in many practical problems the posterior is highly non-Gaussian (quite often multimodal) together with non-linear $\phi(.)$ and $g(.)$. Under such circumstances, sequential Monte Carlo technique based particle filtering algorithm gives us a way to solve this posterior estimation problem. In the next few sections, we shall develop the particle filter starting with Monte Carlo sampling for pdf approximation. We also discuss sequential Monte Carlo method (especially sequential importance sampling) and how it leads to particle filtering.

2.2 Derivation of the Particle Filter

Say, we are trying to compute the expectation w.r.t the joint posterior distribution $p(X_{1:t}|Y_{1:t})$

$$E_{p(X_{1:t}|Y_{1:t})}[f(X_{1:t})] = \int f(X_{1:t})p(X_{1:t}|Y_{1:t})dX_{1:t} = ? \tag{2.4}$$

In order to solve this problem, we have to look into two very important issues. a) Do we have a closed form expression for $p(X_{1:t}|Y_{1:t})$? b) Can we sample from $p(X_{1:t}|Y_{1:t})$? Depending on these two issues we can use three different methods to solve this problem namely, simple Monte Carlo sampling, importance sampling and bayesian importance sampling. The method involving bayesian importance is our main focus as it exactly is what bayesian filtering/particle does. These methods are discussed below (taken from [3]).

2.2.1 Simple Monte Carlo Sampling

Consider the case when we can sample from the joint posterior distribution $p(X_{1:t}|Y_{1:t})$. In that case we can approximate the distribution as a discretized version in terms of the samples drawn from that distribution. Or,

$$\begin{aligned} X_{1:t}^{(i)} &\sim p(X_{1:t}|Y_{1:t}), \quad \text{i=1, ..., N} \\ \hat{p}(X_{1:t}|Y_{1:t}) &\approx \sum_{i=1}^{N} \frac{1}{N}\delta(X_{1:t} - X_{1:t}^{(i)}) \end{aligned} \tag{2.5}$$

Now, we can approximate the expectation in the equation (2.4) as follows,

$$\begin{aligned} E_{p(X_{1:t}|Y_{1:t})}[f(X_{1:t})] &\approx E_{\hat{p}(X_{1:t}|Y_{1:t})}[f(X_{1:t})] \\ &= \frac{1}{N}\sum_{i=1}^{N} f(X_{1:t}^{(i)}) \end{aligned}$$

This is the most ideal case. In reality, almost always we cannot sample from the posterior but we might have a closed form expression for the distribution. This leads to the second method known as importance sampling.

2.2.2 Bayesian Importance Sampling

Importance sampling can be used to estimate the properties of a particular distribution, while only having samples from a different distribution (known as importance density or importance function) rather than the distribution of interest. Thus when we have a closed form expression for $p(X_{1:t}|Y_{1:t})$ but cannot sample from it, we use an importance density to draw the samples [3]. We choose the importance density such that it has a convenient closed form expression and can easily draw samples from it. Say, this importance density be given as $\pi(X_{1:t}|Y_{1:t})$. Now, the problem of posterior expectation computation can be solved as follows,

$$\begin{aligned} I_t &= \int f(X_{1:t})p(X_{1:t}|Y_{1:t})dX_{1:t} \\ &= \int f(X_{1:t})\frac{p(X_{1:t}|Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}\pi(X_{1:t}|Y_{1:t})dX_{1:t} \\ &= E_{\pi(X_{1:t}|Y_{1:t})}[f(X_{1:t})\frac{p(X_{1:t}|Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}] \end{aligned}$$

Now, draw samples from $\pi(.)$ and get its discretized version $\hat{\pi}(.)$. Then the computation of posterior expectation boils down to,

$$\begin{aligned} X_{1:t}^{(i)} &\sim \pi(X_{1:t}|Y_{1:t}), \text{ i=1, ..., N} \\ E_{p(X_{1:t}|Y_{1:t})}[f(X_{1:t})] &\approx E_{\hat{\pi}(X_{1:t}|Y_{1:t})}[f(X_{1:t})\frac{p(X_{1:t}|Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}] \\ &= \frac{1}{N}\sum_{i=1}^{N} f(X_{1:t}^{(i)})\frac{p(X_{1:t}^{(i)}|Y_{1:t})}{\pi(X_{1:t}^{(i)}|Y_{1:t})} \end{aligned} \tag{2.6}$$

The corresponding approximation to the joint posterior distribution is given as follows,

$$\begin{aligned} p(X_{1:t}|Y_{1:t}) &\approx \hat{p}(X_{1:t}|Y_{1:t}) \\ &= \sum_{i=1}^{N} w_t^{(i)}\delta(X_{1:t} - X_{1:t}^{(i)}) \text{ where} \\ w_t^{(i)} &= \frac{1}{N}\tilde{w}_t^{(i)}, \text{ with } \tilde{w}_t^{(i)} = \frac{p(X_{1:t}^{(i)}|Y_{1:t})}{\pi(X_{1:t}^{(i)}|Y_{1:t})} \end{aligned}$$

Thus we can see that even if we cannot sample from the joint posterior, still we can get around the problem by performing importance sampling from a conveniently chosen probability distribution $\pi(.)$. But computation of the weights might be a problem in this case; this takes us to the next section.

However, it turns out that in most real-life situations, we can neither sample from the posterior (i.e. $p(X_{1:t}|Y_{1:t})$) nor we have any closed form expression of it. Thus computing the integral in equation (2.4) becomes challenging. Bayesian importance sampling is a variant of the standard importance sampling technique to tackle this problem. The numerical methods developed for solving this problem leads to sequential importance sampling (SIS) and then finally to particle/bayesian filtering. Now, under the problem statement, we do not have a closed form expression for the posterior $p(X_{1:t}|Y_{1:t})$. But it can be expressed in the following manner,

$$\begin{aligned} p(X_{1:t}|Y_{1:t}) &= \frac{p(X_{1:t}, Y_{1:t})}{p(Y_{1:t})} \text{ thus} \\ p(X_{1:t}|Y_{1:t}) &\propto p(X_{1:t}, Y_{1:t}) \end{aligned} \tag{2.7}$$

It turns out that we can at least compute $p(X_{1:t}, Y_{1:t})$ in closed form recursively. This follows from the hidden Markov model (HMM) assumption on the state space. The recursion can be performed as $p(X_{1:t}, Y_{1:t}) = p(X_{1:t-1}, Y_{1:t-1})p(Y_t|X_t)p(X_t|X_{t-1})$ with $p(Y_t|X_t)$ and $p(X_t|X_{t-1})$ known. We shall utilize this fact to solve the problem of computing the posterior expectation in a recursive fashion i.e. starting with a known $p(x_0)$ and using $p(Y_t|X_t)$, $p(X_t|X_{t-1})$, keep computing the approximate posterior $\hat{p}(X_{1:t}|Y_{1:t})$ and $E_{p(X_{1:t}|Y_{1:t})}[f(X_{1:t})]$ for $t > 0$.

The Bayesian importance sampling is performed as follows [3]. The posterior expectation $E_{p(X_{1:t}|Y_{1:t})}[f(X_{1:t})]$ can be written as,

$$\begin{aligned} I_t &= \int f(X_{1:t})p(X_{1:t}|Y_{1:t})dX_{1:t} \\ &= \int f(X_{1:t})\frac{p(X_{1:t}, Y_{1:t})}{p(Y_{1:t})}dX_{1:t} \\ &= \frac{\int f(X_{1:t})p(X_{1:t}, Y_{1:t})dX_{1:t}}{p(Y_{1:t})} \\ &= \frac{\int f(X_{1:t})p(X_{1:t}, Y_{1:t})dX_{1:t}}{\int_{X_{1:t}} p(X_{1:t}, Y_{1:t})dX_{1:t}} \end{aligned} \tag{2.8}$$

Now, let us consider the importance density to be $\pi(X_{1:t}|Y_{1:t})$ from which we are going to draw samples. We choose $\pi(.)$ in such a way that we at least recursively know how to compute the expression for $\pi(X_{1:t}|Y_{1:t})$. The importance density $\pi(.)$ is assumed to have certain properties which are crucial to the development of the recursive algorithm for approximating

the posterior distribution and expectation. These will be discussed soon. Now, let us get back to the problem of computing I_t. From equation (2.8) it follows that

$$\begin{aligned} I_t &= \frac{\int f(X_{1:t}) \frac{p(X_{1:t},Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})} \pi(X_{1:t}|Y_{1:t}) dX_{1:t}}{\int_{X_{1:t}} \frac{p(X_{1:t},Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})} \pi(X_{1:t}|Y_{1:t}) dX_{1:t}} \\ &= \frac{E_{\pi(.)}[f(X_{1:t}) \frac{p(X_{1:t},Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}]}{E_{\pi(.)}[\frac{p(X_{1:t},Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}]} \end{aligned} \tag{2.9}$$

Now we can sample from $\pi(.)$ as, $X_{1:t}^{(i)} \sim \pi(X_{1:t}|Y_{1:t})$, $i = 1, \ldots, N$ and then get its discretized version $\hat{\pi}(.)$. Finally, I_t can be estimated as follows,

$$\begin{aligned} \hat{I}_t &= \frac{E_{\hat{\pi}(.)}[f(X_{1:t}) \frac{p(X_{1:t},Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}]}{E_{\hat{\pi}(.)}[\frac{p(X_{1:t},Y_{1:t})}{\pi(X_{1:t}|Y_{1:t})}]} \\ &= \frac{\frac{1}{N}\sum_{i=1}^{N} f(X_{1:t}^{(i)}) \tilde{w}_t^{(i)}}{\frac{1}{N}\sum_{j=1}^{N} \tilde{w}_t^{j}}, \quad \tilde{w}_t^{(i)} = \frac{p(X_{1:t}^{(i)}, Y_{1:t})}{\pi(X_{1:t}^{(i)}|Y_{1:t})} \\ &= \sum_{i=1}^{N} f(X_{1:t}^{(i)}) w_t^{(i)}, \text{ with } w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{j}} \end{aligned} \tag{2.10}$$

The corresponding approximation to the joint posterior distribution is given as,

$$\hat{p}(X_{1:t}|Y_{1:t}) \triangleq \sum_{i=1}^{N} w_t^{(i)} \delta(X_{1:t} - X_{1:t}^{(i)}) \tag{2.11}$$

where $\{w_t^{(i)}\}_{\forall i}$ are called the normalized importance weights. Now, clearly w_t^i's cannot be computed directly at a given instant. So, we develop a recursive way to compute them. Once this is done, we have a computationally feasible method for computing the posterior distribution. In order to do that we first make sure the following assumption holds for the importance density $\pi(X_{1:t}|Y_{1:t})$.

$$\pi(X_{1:t}|Y_{1:t}) = \pi(X_{1:t-1}|Y_{1:t-1}) \pi(X_t|X_{1:t-1}, Y_{1:t}) \tag{2.12}$$

Since $p(X_{1:t}, Y_{1:t}) = p(X_{1:t-1}, Y_{1:t-1}) p(Y_t|X_t) p(X_t|X_{t-1})$, we can develop a recursive way of

computing the importance weights as,

$$\begin{aligned} \tilde{w}_t^{(i)} &= \frac{p(X_{1:t}^{(i)}, Y_{1:t})}{\pi(X_{1:t}^{(i)}|Y_{1:t})} \\ &= \tilde{w}_{t-1}^{(i)} \frac{p(Y_t|X_t^{(i)})p(X_t^{(i)}|X_{t-1}^{(i)})}{\pi(X_t^{(i)}|X_{1:t-1}^{(i)}, Y_{1:t})} \text{ where,} \\ X_t^{(i)} &\sim \pi(X_t|X_{1:t-1}^{(i)}, Y_{1:t}) \text{ and } X_{1:t}^{(i)} = [X_{1:t-1}^{(i)}, X_t^{(i)}] \end{aligned} \tag{2.13}$$

Thus we can recursively compute the estimates of the posterior distribution and the corresponding expectations starting with the initial distribution. The entire algorithm is summarized in the next section which is known as sequential importance sampling (SIS).

2.2.3 Sequential Importance Sampling (SIS)

Before we give the algorithm for SIS, it is important to choose the importance density $\pi(.)$. There could be many choices. The simplest one is to use the state transition density as the importance density [2]. Or,

$$\pi(X_t|X_{1:t-1}, Y_{1:t}) = p(X_t|X_{t-1}) \tag{2.14}$$

$$\text{This gives, } \tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)} p(Y_t|X_t^{(i)}) \tag{2.15}$$

The optimal importance density [4, 3] is the one which minimized the variance of the importance weights conditioned on the observations and previous state samples. It can be shown that $\pi_{opt}(.) = p(X_t|X_{t-1}, Y_t)$ under the hidden Markov model (HMM) assumption. Finally, the recursive algorithm for estimating the posterior density is summarized below.

SIS Algorithm

1. Sample from the initial distribution. $X_0^{(i)} \sim p(X_0)$, assign $\tilde{w}_0^{(i)} = w_0^{(i)} = \frac{1}{N}$, $i = 1, ..., N$

2. For $t > 0$ and $i = 1, ..., N$,

(a) Sample $X_t^{(i)} \sim p(X_t|X_{t-1}^{(i)})$ and $X_{1:t}^{(i)} = [X_{1:t-1}^{(i)}, X_t^{(i)}]$. Compute weights as, $\tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)} p(Y_t|X_t^{(i)})$

(b) Get the normalized importance weights $w_t^{(i)}$ and finally, $\hat{p}(X_{1:t}|Y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(X_{1:t} - X_{1:t}^{(i)})$

3. Set $t + 1 \leftarrow t$ and go back to step 2.

The SIS algorithm [3] gives us a way to recursively estimate the discretized posterior distribution. But this algorithm has one major problem which makes it almost ineffective for practical applications. It turns out that the variance of the importance weights conditioned on the observations only increases over time. As a result, after a few iterations, only a few samples (also called *particles*) will have non-zero normalized importance weights. This is known as degeneracy of weights and it ends up wasting a lot of particles making it very inefficient. The particle filter comes into picture to solve this problem. The basic particle filtering algorithm is discussed in the next section.

2.2.4 The Basic Particle Filter

The traditional particle filter [2] basically modifies the SIS algorithms to prevent the degeneracy of weights. It is also important to note that basic PF uses the state transition prior i.e. $p(X_t|X_{t-1})$ as the importance density. As we will see in the next section, this is not the optimal importance density. In order to get around the problem of degeneracy of weights, what the basic PF does is: at each time step, the particles (i.e. the samples) are resampled w.r.t their normalized importance weights. Intuitively, what this step does is : get rid of the unlikely particles and replace them by breeding multiple copies of likely particles i.e. particles with high importance weights. In other words, $\{w_t^{(i)}\}_{i=1}^{N}$ is used as a probability mass function (PMF) to sample the existing particles again. It is denoted as $X_t^{(i)} \sim PMF[\{w_t\}]$ and the new importance weights are assigned as $w_t^{(i)} = \frac{1}{N}$. The basic particle filtering algorithm is summarized below.

Particle Filtering (PF) Algorithm

1. Sample from the initial distribution. $X_0^{(i)} \sim p(X_0)$, assign $\tilde{w}_0^{(i)} = w_0^{(i)} = \frac{1}{N}$, $i = 1, ..., N$
2. For $t > 0$,
   (a) Sample $X_t^{(i)} \sim p(X_t|X_{t-1}^{(i)})$ and Compute weights as, $\tilde{w}_t^{(i)} = p(Y_t|X_t^{(i)})$, $i = 1, ..., N$
   (b) Get the normalized importance weights $w_t^{(i)}$ as $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^j}$
   (c) Resample the particles as, $X_t^{(i)} \sim PMF[\{w_t\}]$ and reassign $w_t^{(i)} = \frac{1}{N}$
   (d) Get the estimated posterior as, $\hat{p}(X_{1:t}|Y_{1:t}) = \sum_{i=1}^{N} \frac{1}{N} \delta(X_{1:t} - X_{1:t}^{(i)})$ where $X_{1:t}^{(i)} = [X_{1:t-1}^{(i)}, X_t^{(i)}]$
   (e) Finally, compute $I_t = E_{p(x_t|Y_{1:t})}[f(X_t)] \approx E_{\hat{p}(x_t|Y_{1:t})}[f(X_t)] = \frac{1}{N} \sum_{i=1}^{N} f(X_t^{(i)})$
3. Set $t + 1 \leftarrow t$ and go back to step 2.

Apart from this basic PF, there are other variants of the PF exist in literature e.g. Auxiliary Particle Filter (Aux-PF) [87], Unscented Particle Filter, Rao-Blackwellized Particle Filter (RB-PF) [3] and PF-Doucet [3]. Some of them tackles the problem of degeneracy e.g. Aux-PF, while some others attempt to improve the effective particle size e.g. PF-Doucet and RB-PF. More discussions about the relevant algorithms to follow.

2.3 Developing Efficient Particle Filters

First, let us restate the problem definition. Our goal is to sequentially estimate (track) a hidden sequence of states, X_t, from a sequence of observations, Y_t, which satisfy the Hidden Markov Model (HMM) property, i.e.

1. For each t, the dependence $X_t \rightarrow Y_t$ is Markovian, with observation likelihood (OL) represented as $p(Y_t|X_t)$.
2. For each t, the dependence $X_{t-1} \rightarrow X_t$ is Markovian, with state transition pdf (STP), $p(X_t|X_{t-1})$.

The posterior, $\pi_t(X_t) \triangleq p(X_t|Y_{1:t})$, needs to be recursively computed at each t, using π_{t-1} and the current observation Y_t. Once the posterior is available, any "optimal" state estimate, e.g. MAP or MMSE, can be computed. As mentioned above, a particle filter (PF) uses sequential importance sampling [3] along with a resampling step [2] to output at each time t, a cloud of N particles, $\{X_t^{(i)}\}$ with weights $\{w_t^{(i)}\}$ whose empirical measure $\pi_{t|t}^N(X_t) \triangleq \sum_{i=1}^N w_t^{(i)} \delta(X_t - X_t^{(i)})$ closely approximates the true posterior, $\pi_{t|t}(X_t) \triangleq p(X_t|Y_{1:t})$. The tracking problem is complicated by the following two facts:

1. For a given observation, Y_t, the OL (as a function of the state, X_t) is often multimodal or heavy tailed. If the STP is broad even in some dimensions, it will result in the posterior given previous state, p^*, defined as

$$p^*(X_t) \triangleq p(X_t|X_{t-1}, Y_t) \tag{2.16}$$

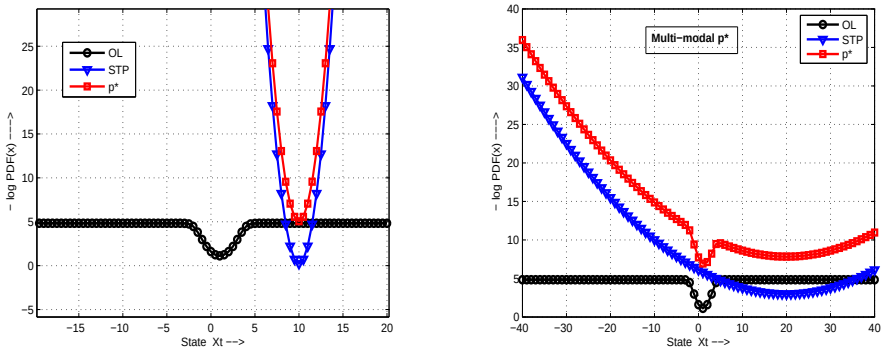
   becomes multimodal. It is demonstrated in Fig. 2.2 and we will have more discussion on this issue later.

2. A second issue is applying the PF when the state space dimension is large. The effective particle size reduces with dimension, thus requiring a larger number of particles for a given accuracy as dimension increases.

Efficient Particle Filters developed in the sections to follow tackles these issues by carefully splitting the state-spacer and using smart importance sampling techniques for individual parts depending on their properties.

2.3.1 PF-EIS : PF with Efficient Importance Sampling

As mentioned earlier, the first PF algorithm, PF-Gordon [2], used the state transition prior (i.e. $p(X_t|X_{t-1})$) as the importance density. This assumes nothing and has very small computation burden per particle. But since it does not use knowledge of the current observation, the weights variance can be large, particularly when the observations are more reliable than the prior model. Thus it requires more particles for a given accuracy level compared to the case



Narrow STP Broad STP

Figure 2.2 Consider a scalar problem with STP $p(X_t|X_{t-1}^i) = \mathcal{N}(X_t; 0.5X_{t-1}^i, \sigma_s^2)$ and OL, $p(Y_t|X_t) = 0.8\mathcal{N}(X_t, \sigma_o^2) + 0.2\mathcal{N}(0, 100\sigma_o^2)$. The OL is a raised Gaussian, i.e. it is heavy tailed with mode at Y_t. Thus whenever Y_t is generated by the outlier component, the OL mode is far from the STP mode. If the STP is broad, as in the right-side plot, this results in a bimodal p^*. We plot the $-\log$ of the OL, STP and p^* for $Y_t = 1$, $\sigma_o^2 = 1$, and $0.5X_{t-1}^i = 10$, $\sigma_s^2 = 0.25$ (narrow) and $0.5X_{t-1}^i = 20$, $\sigma_s^2 = 16$ (broad).

when the knowledge of observations are used. The optimal importance density [3] is given by the posterior conditioned on the previous state, denoted by p^* where,

$$p^*(X_t) \triangleq p(X_t|X_{t-1}, Y_t) \tag{2.17}$$

But in most problems, including ours, p^* cannot be computed analytically. When it is unimodal, PF-Doucet [3] approximates it by a Gaussian about its mode (Laplaces approximation [5]) and samples from the Gaussian. Other work that also implicitly assume that p^* is unimodal includes [4, 20]. In many practical scenarios (e.g. visual tracking under occlusion and clutter), the observation likelihood is a raised Gaussian as a function of X_t, and is thus heavy tailed. If the equivalent state transition prior of X_t is broad, whenever Y_t is generated from the outlier distribution (e.g. from clutter), the resulting posterior given the previous state, $p^*(X_t) \triangleq p(X_t|X_{t-1}, Y_t)$ will be multimodal.

For such problems where p^* is often multimodal, particle filter with efficient importance sampling (PF-EIS) was proposed in [15] which combines the ideas of both PF-Gordon and PF-Doucet to handle multimodal observation likelihoods. This algorithm relies on the fact that

Algorithm 1 PF-EIS. Going from π_{t-1}^N to $\pi_t^N(X_t) = \sum_{i=1}^N w_n^{(i)} \delta(X_t - X_t^{(i)})$, $X_t^{(i)} = [X_{t,s}^{(i)}, X_{t,r}^{(i)}]$

1. *Importance Sample $X_{t,s}$:* $\forall i$, sample $X_{t,s}^{(i)} \sim p(X_{t,s}^{(i)}|X_{t-1}^{(i)})$.
2. *Importance Sample on $X_{t,r}$:* $\forall i$, sample $X_{t,r}^{(i)} \sim \mathcal{N}(m_t^{(i)}, \Sigma_{IS}^{(i)})$ where $m_t^{(i)} = \arg\min_{X_{t,r}} L^{(i)}(X_{t,r})$ and $\Sigma_{IS}^{(i)} = [\nabla^2 L^{(i)}(m_t^{(i)})]^{-1}$ where $L^{(i)}(X_{t,r}) \triangleq -\log(p^{**,(i)}(X_{t,r})) = \log[p(X_{t,r}|X_{t-1}^{(i)}, X_{t,s}^{(i)}, Y_t)]$
3. *Weight:* $\forall i$, compute $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}}$ where $\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{p(Y_t|X_t^{(i)}) p(X_{t,r}^{(i)}|X_{t-1}^{(i)}, X_{t,s}^{(i)})}{\mathcal{N}(X_{t,r}^{(i)}; m_t^{(i)}, \Sigma_{IS}^{(i)})}$
4. *Resample using any standard algorithm [4].* Set $t \leftarrow t+1$ and go to step 1.

even though p^* is multimodal, for most real-life problems it is possible to split the state vector X_t into an "effective basis" $X_{t,s}$ and "residual space" $X_{t,r}$ in such a way that p^*, conditioned on $X_{t,s}$, is unimodal i.e.

$$p^{**,i}(X_{t,r}) \triangleq p^*(X_t|X_{t,s}^i) = p(X_{t,r}|X_{t-1}^i, X_{t,s}^i, Y_t) \tag{2.18}$$

is unimodal. Here, the index i represents the sample from the i^{th} particle. We sample the $X_{t,s}$ particle, $X_{t,s}^i$, from its state transition prior (STP) but use Laplace's approximation [5, 3] to approximate $p^{**,i}$ by a Gaussian and sample $X_{t,r}$ from it. Thus we sample $X_{t,r}^i$ from $\mathcal{N}(m_t^i, \Sigma_{IS}^i)$ where

$$m_t^i = \arg\min_{X_{t,r}} [-\log p^{**,i}(X_{t,r})] = \arg\min_{X_{t,r}} L^i(X_{t,r}) \tag{2.19}$$

$$\Sigma_{IS}^i = [\nabla^2 L^i(m_t^i)]^{-1} \text{ where} \tag{2.20}$$

$$L^i(X_{t,r}) \triangleq [-\log p(Y_t|X_{t,s}^i, X_{t,r})] + [-\log p(X_{t,r}|X_{t-1}^i, X_{t,s}^i)] \tag{2.21}$$

and $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian pdf with mean μ and covariance matrix Σ. As shown in [15], unimodality of $p^{**,i}$ is ensured if the variance of state transition prior (STP) of $X_{t,r}$ is small enough compared to distance between the modes of OL given $X_{t,s}$ in any direction. Even if $X_{t,s}$ is chosen so that this holds for most particles, at most times, the proposed algorithm will work. More details can be found in [22] and [23].

2.3.2 PF-MT and PF-EIS-MT : Posterior Mode Tracking

Now, if the state-space dimensionality is large (10 or more), it makes particle filtering even more challenging because, the effective particle size [3] gets reduced with increased state-space dimension; the number of particles required for reasonable accuracy in estimating the state becomes very large. To get around this problem, Rao Blackwellization (RB-PF) [27, 6] can be used provided the state space model is conditionally linear-Gaussian, or if some states can be vector quantized into a few discrete centers. For many practical problem, neither assumption holds.

But in most large dimensional problems, the state change variance is large in only a few dimensions i.e. the LDSS property [21] holds (at any given time, "most of the state change" occurs in a small number of dimensions, while the change in the rest of the state space is small). This can be exploited [21] to further split $X_{t,r}$ into $[X_{t,r,s}; X_{t,r,r}]$ so that the covariance of the STP of $X_{t,r,r}$ is small enough to ensure that there is little error in approximating the conditional posterior of $X_{t,r,r}$, $p^{**,i}(X_{t,r,r})$, by a Dirac delta function at its mode. We call this the Mode Tracking (MT) approximation of importance sampling (IS), or IS-MT. *When MT is combined with PF-EIS, the resulting algorithm is called PF-EIS-MT.*. For PF-EIS-MT, the generation of the i^{th} particle of the state vector $X_t = [X_{t,s}, X_{t,r,s}, X_{t,r,r}]$ can be summarized as follows,

1. Importance Sample $X_{t,s}$: $\forall i$, sample $X^i_{t,s} \sim p(X^i_{t,s}|X^i_{t-1})$.

2. Efficient Importance Sample $X_{t,r,s}$: $\forall i$,

   (a) Compute m^i_t and Σ^i_{IS} using (4.11), (2.20). Let $m^i_t = [m^i_{t,s}\ m^i_{t,r}]$ and $\Sigma^i_{IS} = \begin{bmatrix} \Sigma_{IS,s}\ \Sigma_{IS,s,r} \\ \Sigma_{IS,r}\ \Sigma_{IS,r,s} \end{bmatrix}$.

   (b) Sample $X^i_{t,r,s} \sim \mathcal{N}(m^i_{t,s},\ \Sigma^i_{IS,s})$.

3. Mode Track $X_{t,r,r}$: $\forall i$, set $X^i_{t,r,r} = m^i_{t,r} + \Sigma^i_{IS,r,s}(\Sigma^i_{IS,s})^{-1}(X^i_{t,r,s} - m^i_{t,s})$

The PF-EIS-MT algorithm is given in Algorithm. 2.

Algorithm 2 PF-EIS-MT. Going from π_{t-1}^N to $\pi_t^N(X_t) = \sum_{i=1}^N w_n^{(i)} \delta(X_t - X_t^i)$, $X_t^i = [X_{t,s}^i, X_{t,r}^i]$, $X_{t,r}^i = [X_{t,r,s}^i, X_{t,r,r}^i]$

1. *Importance Sample $X_{t,s}$:* $\forall i$, sample $X_{t,s}^i \sim p(X_{t,s}^i | X_{t-1}^i)$.
2. *Efficient Importance Sample $X_{t,r,s}$:* $\forall i$,
   (a) Compute m_t^i and Σ_{IS}^i using (4.11), (2.20). Let $m_t^i = [m_{t,s}^i \; m_{t,r}^i]$ and $\Sigma_{IS}^i = \begin{bmatrix} \Sigma_{IS,s} & \Sigma_{IS,s,r} \\ \Sigma_{IS,r} & \Sigma_{IS,r,s} \end{bmatrix}$.
   (b) Sample $X_{t,r,s}^i \sim \mathcal{N}(m_{t,s}^i, \; \Sigma_{IS,s}^i)$.
3. *Mode Track $X_{t,r,r}$:* $\forall i$, set $X_{t,r,r}^i = m_{t,r}^i + \Sigma_{IS,r,s}^i (\Sigma_{IS,s}^i)^{-1} (X_{t,r,s}^i - m_{t,s}^i)$
4. *Weight:* $\forall i$, compute $w_t^i = \frac{\tilde{w}_t^i}{\sum_{j=1}^N \tilde{w}_t^j}$ where $\tilde{w}_t^i = w_{t-1}^i \frac{p(Y_t|X_t^i) p(X_{t,r}^i | X_{t-1}^i, X_{t,s}^i)}{\mathcal{N}(X_{t,r}^i; \, m_t^i, \, \Sigma_{IS}^i)}$ where $X_{t,r}^i = [X_{t,r,s}^i, X_{t,r,r}^i]$.
5. *Resample using any standard algorithm [4].* Set $t \leftarrow t+1$ & go to step 1.

The IS-MT approximation introduces some error in the estimate of $X_{t,r,r}$ (error decreases with decreasing spread of $p^{**,i}(X_{t,r,r})$). But it also reduces the importance sampling dimension from $\dim(X_t)$ to $\dim([X_{t,s}; X_{t,r,s}])$ (a significant reduction for large dimensional problems), thus improving the effective particle size. For carefully chosen dimension of $[X_{t,s}; X_{t,r,s}]$, this results in smaller total error, especially when the available number of particles, N, is small. This is observed experimentally, but proving it theoretically is an open problem for future research.

A computationally simpler modification of PF-EIS-MT is PF-MT which was developed in earlier work [21]. In PF-MT, we combine $X_{t,r,s}$ with $X_{t,s}$ and importance sample the combined state $\tilde{X}_{t,s} = [X_{t,s}, X_{t,r,s}]$ from its STP, while performing mode tracking (MT) on $\tilde{X}_{t,r} = X_{t,r,r}$. To put the PF-MT idea in a much simpler fashion we include the following section.

2.3.2.1 PF-MT : Basic Idea

As mentioned earlier, PF-MT splits the state vector X_t into a small dimensional "effective basis" (i.e. $X_{t,s}$ in which most of the state change is assumed to occur) and the rest of the state vector, $X_{t,r}$, belonging to the "residual space" in which the state change is assumed "small".

Algorithm 3 PF-MT. Going from π_{t-1}^N to $\pi_t^N(X_t) = \sum_{i=1}^N w_n^{(i)} \delta(X_t - X_t^{(i)})$, $X_t^{(i)} = [X_{t,s}^{(i)}, X_{t,r}^{(i)}]$

1. *Importance Sample $X_{t,s}$:* $\forall i$, sample $X_{t,s}^{(i)} \sim p(X_{t,s}^{(i)}|X_{t-1}^{(i)})$.
2. *Mode Tracking on $X_{t,r}$:* $\forall i$, Compute the mode $m_t^{(i)}$ of $p^{**,i}(X_{t,r})$ and set $X_{t,r}^{(i)} = m_t^{(i)}$ (see equation(2.22))
3. *Weight:* $\forall i$, compute $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}}$ where $\tilde{w}_t^{(i)} = w_{t-1}^{(i)} p(Y_t|X_t^{(i)}) p(X_{t,r}^{(i)}|X_{t-1}^{(i)}, X_{t,s}^{(i)})$
4. *Resample using any standard algorithm [4].* Set $t \leftarrow t+1$ and go to step 1.

It importance samples only on the effective basis dimensions, but replace importance sampling by deterministic posterior Mode Tracking (MT) in the residual space. *Thus the importance sampling dimension is only $dim(X_{t,s})$ (much smaller than $dim(X_t)$) and this is what decides the effective particle size.*

To be precise, PF-MT assumes that the total variance of state change in residual space, $trace(\Sigma_r)$, where Σ_r denotes covariance of the state transition prior (STP) of $X_{t,r}$, is small enough to satisfy the following two assumptions for most particles at most times:

1. The conditional posterior in residual space (posterior of $X_{t,r}$ given $X_{t,s}^i$, $X_{t-1,r}^{(i)}$), denoted $p^{**,i}$, is unimodal. $p^{**,i}$ is defined as

$$\begin{aligned} p^{**,i}(X_{t,r}) &\triangleq p(X_{t,r}|X_{t,s}^{(i)}, X_{t-1}^{(i)}, Y_t) \\ &\propto \overbrace{p(Y_t|X_{t,s}^i, X_{t,r})}^{OL} \overbrace{p(X_{t,r}|X_{t-1,r}^i)}^{STP\ of\ X_{t,r}} \end{aligned} \tag{2.22}$$

2. The sum of eigenvalues of Σ_r, $trace(\Sigma_r)$ is "small enough" to justify the Importance Sampling with Mode Tracking (IS-MT) approximation explained below.

When only Assumption 1 holds, one can use the following importance sampling strategy (PF-EIS, [15]): sample $X_{t,s}^{(i)}$, from its state transition prior, $p(X_{t,s}|X_{t-1,s}^{(i)})$, but sample $X_{t,r}^{(i)}$ from a Gaussian approximation [3], denoted $\mathcal{N}(m_t^{(i)}, \Sigma_{IS}^{(i)})$, to $p^{**,i}$ about its mode, denoted $m_t^{(i)}$. The mode $m_t^{(i)}$ is computed as the minimizer of $L^{(i)}(X_{t,r}) = -\log p^{**,i}(X_{t,r})$ and $\Sigma_{IS}^{(i)} = (\nabla^2 L^{(i)}(m_t^{(i)}))^{-1}$. It is easy to see that [15] $\Sigma_{IS}^{(i)} \leq \Sigma_r$.

If both the Assumptions hold, we can replace importance sampling (IS) from $\mathcal{N}(m_t^{(i)}, \Sigma_{IS}^{(i)})$ (EIS) by deterministically setting $X_{t,r}^{(i)} = m_t^{(i)}$. This is the Mode Tracking (MT) approximation of importance sampling(IS) or IS-MT. The resulting algorithm is called PF-MT (see Algorithm. 3).

Thus, in summary, PF-MT can be used if the variance of residual state change, trace(Σ_r), is small enough in comparison to the distance between the observation likelihood modes and is also small enough to justify the IS-MT approximation.

2.4 Performance Comparison : Simple Static Case

Consider the problem of estimating spatially varying temperature (temperature field) from a network of sensors, which obtain noisy observations of temperature and some of them could occasionally fail. Assume that we have sensors $S_1, \ldots, S_K$ in K different spatial locations. The corresponding true temperature is $C = [C_1, \ldots, C_K]^T$ and the sensor observations are $Y = [Y_1, \ldots, Y_K]^T$. Define $V \triangleq [V_1, \ldots, V_K]^T$ where, V_i is the coefficient along the i^{th} eigen direction of temperature variation. The relationship between C and V is given as, $C = m_c + BV$, where m_c is the mean temperature vector and B is a $K \times K$ orthonormal matrix with its columns as the eigen directions of temperature variation. Thus the state vector becomes, $X = [C^T, V^T]^T$. The prior on V is given as, $p(V) = \mathcal{N}(V; \mathbf{0}, \Sigma_v)$.

We assume that any sensor fails with probability $(1 - p)$ independent of all other sensors. When the sensor is working properly, the observation is a noise-corrupted scaled version of the original temperature. But when the sensor fails, the observation is independent of the true temperature at the sensor location. We model it as a large variance Gaussian. To summarize, the observation likelihood (OL) is given as follows:

$$p(Y|X) = p(Y|C) = \prod_{i=1}^{K} [p\mathcal{N}(\alpha_o C_i, \sigma_o^2) + (1-p)\mathcal{N}(\mathbf{0}, 10\sigma_o^2)] \tag{2.23}$$

where α_o is a scaling factor and σ_o^2 is the observation noise variance. Since C is deterministic given V, we performed importance sampling on V and computed $C = m_c + BV$.

We simulated the above system with $K = 7$ sensors, $p = 0.8$, $\alpha_o = 0.9$, $\sigma_o = 0.5$, $m_c =$

Sl no.	Importance Sampling method ($N = 30$)	Avg. Norm. RMSE
1	EIS-MT ($X_s = [V_1]$, $X_{r,s} = [V_2, V_3]$, $X_{r,r} = [V_4, V_5, V_6, V_7]$)	**0.0416**
2	EIS-MT ($X_s = [V_1, V_2]$, $X_{r,s} = [V_3, V_4]$, $X_{r,r} = [V_5, V_6, V_7]$)	0.0593
3	EIS ($X_s = [V_1]$, $X_{r,s} = [V_2, V_3, V_4, V_5, V_6, V_7]$, $X_{r,r} = empty$)	0.0449
4	IS-Gaussian ($X_s = empty$, $X_{r,s} = [V]$, $X_{r,r} = empty$)	0.0610
5	IS-prior ($X_s = [V]$, $X_{r,s} = empty$, $X_{r,r} = empty$)	0.0733

Sl no.	Importance Sampling method ($N = 100$)	Avg. Norm. RMSE
1	EIS-MT ($X_s = [V_1]$, $X_{r,s} = [V_2, V_3]$, $X_{r,r} = [V_4, V_5, V_6, V_7]$)	0.0375
2	EIS-MT ($X_s = [V_1, V_2]$, $X_{r,s} = [V_3, V_4]$, $X_{r,r} = [V_5, V_6, V_7]$)	0.0420
3	EIS ($X_s = [V_1]$, $X_{r,s} = [V_2, V_3, V_4, V_5, V_6, V_7]$, $X_{r,r} = empty$)	**0.0368**
4	IS-Gaussian ($X_s = empty$, $X_{r,s} = [V]$, $X_{r,r} = empty$)	0.0587
5	IS-prior ($X_s = [V]$, $X_{r,s} = empty$, $X_{r,r} = empty$)	0.0599

Figure 2.3 Comparing EIS-MT with EIS, IS-prior and IS-Gaussian for $N = 30$ (top) and $N = 100$ (bottom)

$[25, ..., 25]^T$ and $\Sigma_v = diag([3^2, 5^2, 2^2, 2^2, 1, 1, 1])$ where $diag(a)$ denotes a diagonal matrix with a as its diagonal. The performance measure of the system is given by averaging the normalized RMSE, $NE = \frac{||C-\hat{C}||}{||C||}$ over 50 Monte Carlo simulations. Here, $\hat{C}$ is the importance sampling estimate of $\mathbb{E}[C|Y]$.

We computed $\hat{C}$ using the following IS techniques and compared the NE values: EIS, EIS-MT, IS-prior and IS-Gaussian-approx. Notice that IS-prior can be interpreted as EIS-MT with $X_s = X$, while IS-Gaussian can be interpreted as EIS-MT with $X_{r,s} = X$. We used two different values of the sample size, $N = 30$ and $N = 100$. Also, while performing EIS-MT we tried two different case : 1) when $X_s = [V_1]$, $X_{r,s} = [V_2, V_3]$, $X_{r,r} = [V_4, V_5, V_6, V_7]$ and 2) when $X_s = [V_1, V_2]$, $X_{r,s} = [V_3, V_4]$, $X_{r,r} = [V_5, V_6, V_7]$. The results are summarized in Fig. 2.3. Notice that both EIS and EIS-MT significantly outperform IS-prior and IS-Gaussian. When N is large, EIS has the best performance. But as explained earlier, when N is small, EIS-MT outperforms EIS and and all other methods. This is because in EIS-MT we importance sample only on 3 dimensions (while computing conditional posterior mode for the rest) and thus its effective sample size is much larger.

2.5 Summary

In this chapter, we have given an elaborate description of the development of efficient particle filters which are designed to tackle the challenges associated with large dimensional state space and multimodal observation likelihood. Here, we only give a simple quantitative performance comparison among EIS, EIS-MT and some other methods for a simple static case. Further performance comparisons and demonstration of their practical applications in computer vision can be found in the next two chapters.

CHAPTER 3. Dynamical Models for Landmark Shape Change

In this chapter, we first give a brief summary of our research contributions and related works. Then we develop dynamical models for landmark shape deformations and demonstrate their applications. These include various computer vision problems like tracking various motion activities from videos, automatic landmark extraction, abnormality detection to name a few.

3.1 Contribution and Related Works

The key *contribution* of this work [24, 11] is a novel approach to define a generative model for 2D and 3D nonstationary landmark shape sequences [1]. The main idea is to compute the tangent space representation of the current shape in the tangent space at the previous shape. This can be referred to as the "shape velocity" vector since it quantifies the "difference" between two consecutive shapes projected into the tangent space at the first one. The coefficients of shape velocity along the orthogonal basis directions spanning the current tangent space ("shape speed") can be modeled using standard vector time series models. An important requirement in doing this is to ensure that the basis directions of the current tangent space are *aligned* with those of the previous one. For both 2D and 3D shape sequences, we use the tangent space projections defined in [28, pages 71-77].

A second *contribution* of our work is demonstrating the use of our nonstationary model for (a) sequentially filtering noise-corrupted landmark configurations to compute Minimum Mean Procrustes Square Error (MMPSE) estimates of the true shape; (b) for tracking, i.e. for using the filtering to predict the locations of the landmarks at the current time and using this predic-

[1] We could have just defined the model for m-D landmark shape sequences and 2D or 3D would follow as special cases. But, we model the 2D case separately since both shape computation from preshapes (compare (3.2) versus (3.17)) and Procrustes mean computation is more efficient in 2D than in general m-D (where the mean is computed using an iterative algorithm) [28].

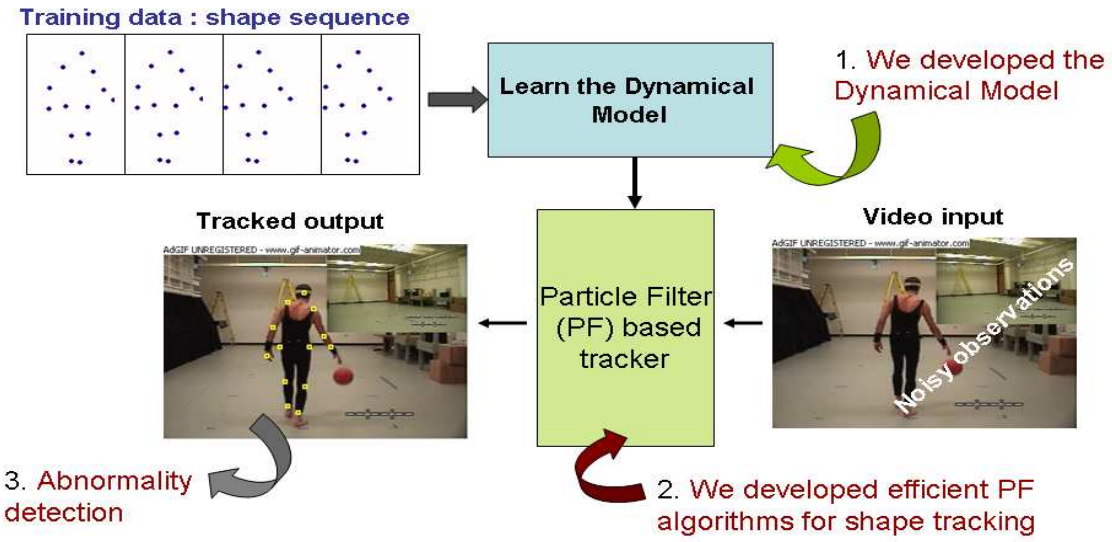


Figure 3.1 The overall landmark shape tracking system is demonstrated. The particle filter based tracker uses the NSSA dynamical model to extract true landmark locations out of observations form noisy images.

tion for faster and more accurate landmarks' extraction from the current image; (overall system is shown in Fig. 3.1) (c) for synthesis; (d) change detection. Greatly improved performance of our tracking and filtering algorithm over existing work [29, 8, 12] is demonstrated. Due to the nonlinearities in the shape dynamics model and the non-Gaussian observation model (similar to that of Condensation [13]), we use a particle filter (PF) [2] for filtering and tracking. Our tracking problem is a typical example of a large dimensional problem with frequently multi-modal observation likelihoods (due to background clutter and missing landmarks) and hence we replace the basic PF used in previous work by the recently proposed PF with Efficient Importance Sampling (PF-EIS). We demonstrate that PF-EIS has a much better performance for landmark shape tracking than the basic PF, when the number of particles used is small.

In recent years, there has been a large amount of work on modeling sequences of landmark shapes - both in statistics [28, 30] and in computer vision and medical image analysis [29, 8, 31, 32, 13, 33, 34, 35, 36]. Active shape models (ASM) [29] and stationary shape activities (SSA) [8] both assume stationarity of the shape sequence (single mean shape plus stationary deviations about it).

But in most real applications, there is large shape variation over a long sequence and so

a single mean shape plus an ASM or SSA model does not suffice. This is explained in more detail in Sec. 3.2.2. For e.g., consider a running sequence (see Fig. 3.21). Another example is the changes in shape within a single heart cycle. In existing work, the ASM is usually replaced by piecewise ASMs [31], for example different ASMs are used for systolic and diastolic motions in [31] or SSA is replaced by piecewise SSA [32]. Piecewise ASMs are good for recognition problems, but not for automatic tracking or for compression since they do not model the transitions between pieces well. When piecewise SSA was used for tracking in [32], it needed to use separate change detection and shape recognition procedures to detect when and which piece to switch to. In this work, we demonstrate through extensive experiments that both filtering and tracking using our model significantly outperforms either ASMs or SSAs.

Smoothing splines [37, 30] is, to the best of our knowledge, the only other existing work that truly models nonstationary landmark shape sequences (other than the piecewise models discussed above). But it does not provide a generative model for the shape sequences, which is the key requirement in tracking, compression or synthesis applications.

The key difference of our work from Condensation [13] is that Condensation only models and tracks global affine deformation between two landmark configurations. This is a valid model for rigid or approximately rigid object motion in front of an affine camera, but not for modeling shape change of different parts of the human body performing different activities/actions such as running or jumping or for activities of groups of interacting persons/vehicles (modeled as landmarks) where there is significant local shape deformation which is not affine.

Our modeling approach is similar in spirit to [38] which also uses piecewise geodesic priors to define a generative model but in a very different context.

Other related work includes Active Appearance Models [33] and Active Appearance Motion Models [34], which also model appearance and hence are not invariant to intensity changes between training and test data, and work on articulated human body tracking [35, 36, 39].

One important computer vision application to our work has been to extend the NSSA model and automatic landmark extraction technique to detect changes and abnormal activities in video sequences. This has been done using the ELL (Expected Log-likelihood) based

change detection statistics defined in [40]. As long as the PF tracking error is under a certain threshold, the ELL statistics can give us a quantitative measure as how badly the observations are following the current system model. Under this setup, if at time t, ELL has exceeded its threshold but the tracking error is still below its threshold (PF is still in track), we declare a change in activity. This can also serve as abnormal activity detection. In case PF looses track, the exploding tracking error can help us detect changes. It makes sense because PF performance is certain to go down as the underlying state sequence no longer follows the system model that the PF relies on. More details on change and abnormality detection can be found in [40, 8, 41].

Another contribution of our work is that we have exploited the dynamical model for 2D/3D landmark shapes in order to develop a novel lossy compression scheme for the landmark shape data extracted from video/volume image sequences. This technique has potential applications in the compact storage of large volumes of biomedical landmarks' data. There are multiple applications where key landmarks of interest are extracted either manually (e.g. by doctors/radiologists in medical imaging applications) or using marker based motion capture technologies (e.g. these are used for human joint motion understanding for biomechanics applications). An example is, the CMU motion capture database ([10], http://mocap.cs.cmu.edu). Now, the question is: Can we model the correlation between temporal landmark shape sequences and use the model for efficient lossy-compression to efficiently reduce the amount of data to be stored? If we can, then it would be a very efficient way of storing large volumes of biomedical landmarks' data. Related work addressing similar questions is [42]. We have used the NSSA model in order to develop the compression scheme for both 2D and 3D landmarks data (details in Sec. 3.5.1 and Sec. 3.5.2). We have compared the compression performances of SSA and ASM with that of NSSA. It was found that NSSA had a better average performance compared to both SSA and ASM. This can again be attributed to the fact that SSA and ASM models tend to break down under large shape variations (e.g. a crawling or a dancing sequence or a running sequence) because of their single mean shape assumptions. On the other hand, NSSA, being a more generic model is less prone to such issues.

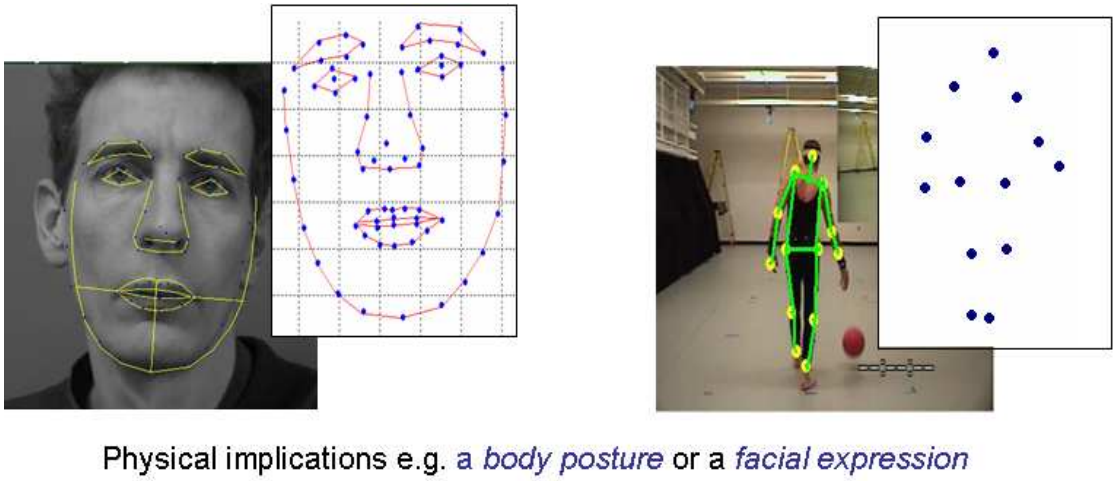


Figure 3.2 Landmark shapes. It could represent anything from a facial expression to a body posture. When the body performs some activity, the corresponding shape deforms in a certain pattern.

In recent years, there has been a significant amount of work on model based video and shape compression. Quite a few of them are in the field of biomedical imaging [42, 43, 44]. For the shape coding in object-based video sequence, [45] uses a context based arithmetic coding of 2D shape sequences. A low bit-rate video compression technique utilizing compact encoding of motion fields has been proposed in [46]. A lossless and near-lossless compression scheme for 4D volume biomedical image sequences has been proposed in [42].

In the next few sections, we develop dynamical models for landmark shape deformations and demonstrate their applications.

3.2 Modeling 2D Shape Sequences

The landmarks are *points of interest* for describing the shape of an object. Two examples are shown in Fig. 3.2. For modeling human motion activity or any activity involving multiple interacting objects, we represent body joints/objects as the landmark points and the corresponding activity is represented as a sequence of deforming landmark shapes over time. It is done two steps. First, we transform the shape sequence to a vector time series using the nonstationary shape deformation model. Then we fit standard statistical models to the time series.

3.2.1 2D Landmark Shape Analysis Preliminaries

We use a discrete representation of shape of a group of K **landmarks**. The various moving objects (point objects) in a group activity or the rigid parts of human body in an action form the "landmarks" as shown in Fig. 3.2. The **configuration** is the set of landmarks, in the 2D case it is the x and y coordinates of the landmarks which can be represented as a K dimensional complex vector [28].

The raw configuration is denoted as S. It can be normalized for translation (moving origin to the centroid of the configuration) and then for scale (normalizing the translation normalized vector by its Euclidean norm) to yield the **pre-shape**, denoted by w. The configuration of K points after translation normalization, denoted by y, lies in $\mathcal{C}^{K-1}$ ((K-1)-dimensional complex space) while the pre-shape, w, lies on a hyper-sphere in $\mathcal{C}^{K-1}$. The shape space can be visualized as shown in Fig. 3.3. A pre-shape w_1 can be aligned with another pre-shape w_0 by finding the rotation angle for the best fit (minimum mean square error fit) and this gives the **Procrustes fit** of w_1 onto w_0 [28]. This is the **shape** of w_1 w.r.t. w_0. The **Procrustes distance** between preshapes w_1 and w_0 is the Euclidean distance between the Procrustes fit of w_1 onto w_0. The **Procrustes mean** of a set of preshapes $\{w_i\}_{i=1}^{N}$ is the minimizer of the sum of squares of Procrustes distances from each w_i to an unknown unit size mean configuration μ [28]. As shown in [28], μ can be computed as the principal eigenvector of the pre-shape covariance matrix, i.e.

$$\mu = \arg \max_{u:||u||=1} u^*[\frac{1}{N}\sum_{i=1}^{N} w_i w_i^*]u \tag{3.1}$$

Any pre-shape of the set can then be aligned w.r.t. this procrustes mean to return the **shape** (denoted by z) w.r.t. the procrustes mean shape, μ [28].

The shape space, $\mathcal{M}$, is a manifold in $\mathcal{C}^{K-1}$ and hence its actual dimension is $\mathcal{C}^{K-2}$. Thus the tangent plane at any point of the shape space is a $\mathcal{C}^{K-2}$ dimensional hyperplane in $\mathcal{C}^{K}$ [28]. The projection of a configuration, S, in the tangent space at a pole, μ, is evaluated [28]

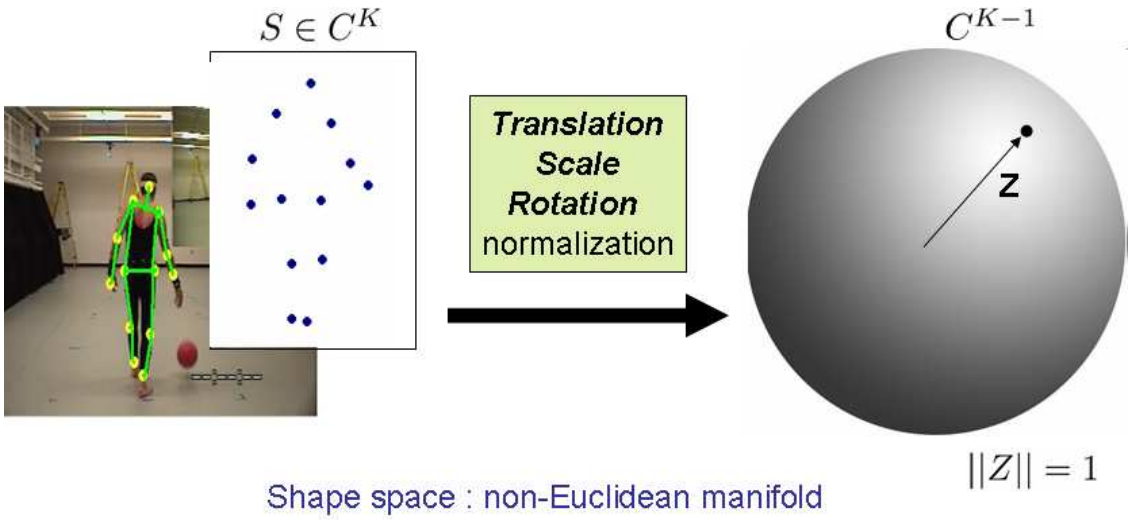


Figure 3.3 Mapping from landmark configuration to a point on the shape space, which is a unit complex hypersphere. The space space being the surface of a sphere is essentially a non-Euclidean manifold.

as follows:

$$\begin{aligned} y &= C_K S, \;\; C_K \triangleq I_K - 1_K 1_K^T / K \\ w &= y/||y|| \\ \theta &= angle(w^* \mu), \;\; z = we^{j\theta} \end{aligned} \tag{3.2}$$

$$v(z, \mu) = [I_K - \mu\mu^T]z \tag{3.3}$$

Here, I_K is a $K \times K$ identity matrix and 1_K is a column vector with K rows with all entries as 1. The notation x^T denotes transpose for real vector x and x^* denotes conjugate transpose for complex vector x. The inverse map (projection from tangent space to shape space) is given by [28],

$$z = (1 - v^* v)^{\frac{1}{2}} \mu + v \tag{3.4}$$

Now, various motion activities will trace out distinctive trajectories (a sequence of z's) on the shape space. Our goal is to model various properties (e.g. velocity etc.) along that trajectory to develop dynamical models for the motion activities. The idea is demonstrated in Fig. 3.4.

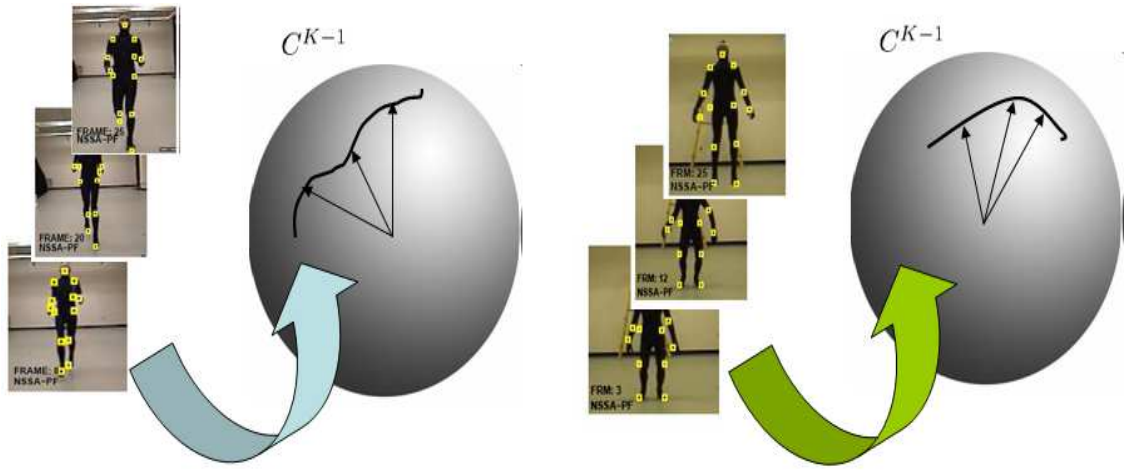


Figure 3.4 This demonstrates the idea that depending on motion activity patterns, corresponding landmark shape sequences will trace out their *signature* trajectories on the shape space.

3.2.2 Problem with SSA and ASM models

The Stationary Shape Activity (SSA) model proposed in [8] computed a single mean shape μ for a training sequence and aligned each preshape, w_t, in the sequence to μ to obtain the shape sequence z_t. Tangent projections, $v(z_t, \mu)$, of each z_t were computed in the tangent space at μ and their time series was modeled using an autoregressive (AR) model. The work of [12] replaced AR by ARMA models and used the models for recognition problems. The Active Shape Model (ASM) of [29] assumed that z_t belongs to a vector space and replaced the tangent space projection given in (3.3) by its linear version $v(z_t, \mu) = z_t - \mu$ and modeled the time series of $v(z_t, \mu)$.

Since both SSA and ASM assumed a single mean shape, they could model only small deviations from mean, which is only possible for stationary sequences. But in many applications, this assumptions may not hold, for example, a crawling or a dancing sequence or see Fig. 3.21. In these cases, the mean shapes for different time intervals are different. Or in other words, considering the entire sequence, the shape activity is essentially nonstationary. Now, if we force a fixed mean shape to such a deforming shape sequence, the resulting shapes, z_t, would drift too far away from μ. It is important to note that a single tangent space approximation works as long as each element of $v(z_t, \mu)$ for all shapes is less than 1 (otherwise the square root

in (3.4) will be of a negative number). Also a time-invariant AR or ARMA model on $v(z_t, \mu)$'s is a valid one only if the magnitudes of each element of $v(z_t, \mu)$ are significantly smaller than 1 (this is because when $v(z_t, \mu)$ is large, i.e. when z_t is far from μ, small changes in $v(z_t, \mu)$ would correspond to very large changes in z_t). But for large shape variation, $v(z_t, \mu)$ will be large. In such a scenario, both SSA and ASM would fail to correctly model the shape dynamics. An intuitive idea about stationary and nonstationary shape sequences is demonstrated in Fig. 3.5.

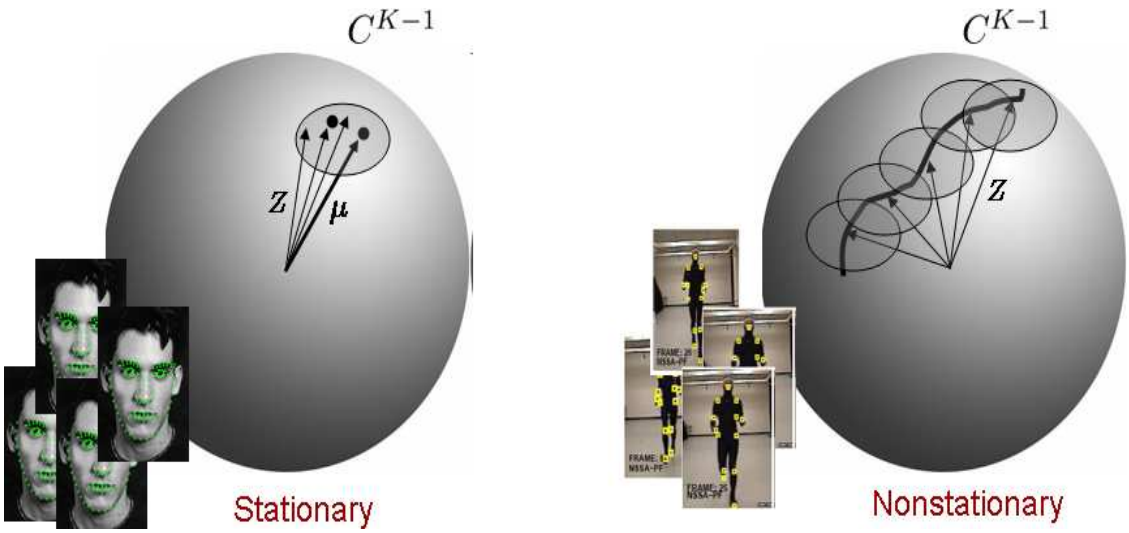


Figure 3.5 Intuitive idea of stationary and nonstationary shape sequences. Stationary shape sequences have single mean shape whereas nonstationary shape sequences can be thought to have evolved form a sequence of time varying mean shapes

3.2.3 Modeling Nonstationary Shape Sequences

To model a nonstationary shape sequence, we use $\mu = z_{t-1}$ at time t. Thus,

$$\begin{aligned} z_t &:= w_t \frac{w_t^* z_{t-1}}{|w_t^* z_{t-1}|} \\ v_t &:= v(z_t, z_{t-1}) = [I - z_{t-1} z_{t-1}^*] z_t \end{aligned} \tag{3.5}$$

The inverse map is given by

$$z_t = (1 - v_t^* v_t)^{\frac{1}{2}} z_{t-1} + v_t \tag{3.6}$$

Since the projection of z_{t-1} in the tangent space at z_{t-1}, $T_{z_{t-1}}$, is zero, v_t can be interpreted as the difference, $(z_t - z_{t-1})$, projected into $T_{z_{t-1}}$, i.e. it is the "shape velocity" at time t. The idea is demonstrated in Fig. 3.6.

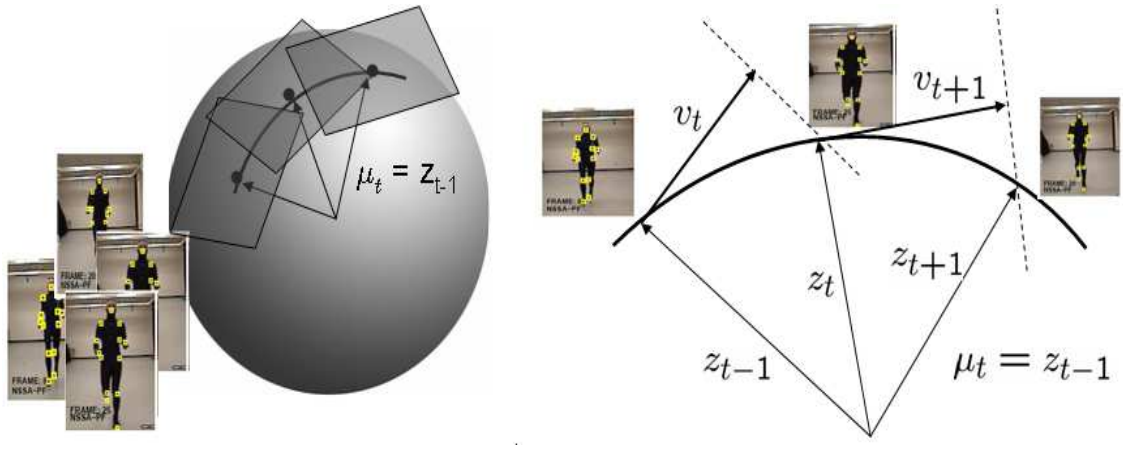


Figure 3.6 The time varying mean shape and 'shape velocity' idea for modeling nonstationary shape sequences.

The translation, scale and rotation normalization in 2D removes 2 complex dimensions (4 real dimensions) and thus the shape space is a $K-2$ dimensional manifold in $\mathcal{C}^K$ and so the tangent space is a $K-2$ dimensional hyperplane in $\mathcal{C}^K$ [28]. Thus the shape velocity, v_t has only $K-2$ independent complex dimensions, i.e. it can be rewritten as $v_t = U_t c_t'$ where the columns of $(U_t)_{K\times K-2}$ contain the $K-2$ orthonormal basis directions spanning $T_{z_{t-1}}$ and $c_t' \in \mathcal{C}^{K-2}$ are the basis coefficients. c_t' may be interpreted as a "shape speed" vector.

Note that, by definition, $T_{z_{t-1}}$ is perpendicular to z_{t-1} and to 1_K. Also, z_{t-1} is perpendicular to 1_K (due to translation normalization). Thus the projection matrix for $T_{z_{t-1}}$ is $[I_K - z_{t-1}z_{t-1}^*]C_K = [I_K - z_{t-1}z_{t-1}^* - 1_K 1_K^T/K]$. In other words, U_t satisfies

$$U_t U_t^* = [I_K - z_{t-1}z_{t-1}^* - 1_K 1_K^T/K] \tag{3.7}$$

One way to obtain U_t is by computing the Singular Value Decomposition (SVD) of the right hand side (RHS) of (3.7) and setting the columns of U_t equal to the left singular vectors with nonzero singular values. Denote this operation by $U_t = left.singular.vectors(M(z_{t-1}))$

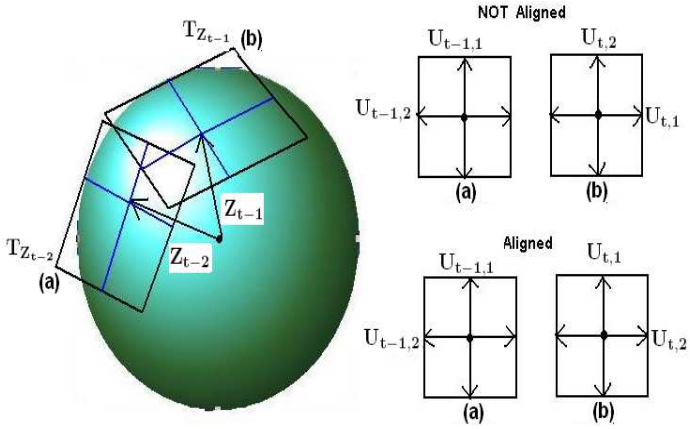


Figure 3.7 This figure shows the alignment of successive tangent spaces for NSSA. When using [8, 41], the axes (here x and y) of the consecutive tangent planes may not be aligned (top). Our method gives aligned axes (bottom).

where,

$$M(z) \triangleq [I_K - zz^* - 1_K 1_K^T/K] \tag{3.8}$$

This was used in [41, 8]. But if this is done at each t, the columns of of U_t and U_{t-1} may not be aligned. As an extreme example consider the following. Let $K = 4$. It may happen that $U_{t-1} = \begin{bmatrix} 1\,0 \\ 0\,1 \\ 0\,0 \\ 0\,0 \end{bmatrix}$ and $U_t = \begin{bmatrix} 0.1 & 0.995 \\ 0.995 & -0.1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$. In this case, it is obvious that the first column of U_t corresponds to the second column of U_{t-1} and vice versa for second column of U_t. Or, in other words, $c'_{t,1}$ corresponds to $c'_{t-1,2}$ and $c'_{t,2}$ to $c'_{t-1,1}$. Thus if SVD is used to obtain U_t at each t, the c'_t's cannot be assumed to be identically distributed and so it is incorrect to model them by an AR model, which assumes stationarity of c'_t. Notice the large modeling error of this method (NSSA-unaligned) in Fig. 3.15(a).

We fix this problem as follows (also see Fig. 3.7). To obtain an aligned sequence of basis directions over time, we obtain the m^{th} column of U_t by starting with the m^{th} column of U_{t-1}, making it perpendicular to z_{t-1} (by subtracting $z_{t-1}z^*_{t-1}$) and then using Gram-Schmidt orthogonalization to also make the resulting vector perpendicular to the first $m-1$ columns of

U_t (i.e. by further subtracting out $\sum_{j=1}^{m-1}(U_t)_j(U_t)_j^*$). This procedure can be summarized as:

$$\begin{aligned} U_t &= g(U_{t-1}, z_{t-1}),\ where \\ g(.)_m &\triangleq [I - z_{t-1}z_{t-1}^* - \sum_{j=1}^{m-1} g(.)_j g(.)_j^*](U_{t-1})_m, \\ &\forall m = 1, \ldots (K-2) \end{aligned} \tag{3.9}$$

Here, $g(.)_m$ denotes the m^{th} columns of $g(U_{t-1}, z_{t-1})$. U_0 is initialized as $U_0 =$ left singular vectors of $M(z_0)$.

Now, since the columns of U_t are aligned, it is fair to assume that $c'_{t,j}$'s are identically distributed for each j over time. Since they are also temporally correlated, we model them by an autoregressive model with lag 1 (AR(1) model). For simplicity of notation, we first convert c'_t into a $2K-4$ dimensional real vector. We denote this operation by,

$$c_t = vec(c'_t) \tag{3.10}$$

and the inverse operation (obtaining the complex vector) is denoted by $c'_t = vec^{-1}(c_t)$. Thus, in summary, the dynamical model of the state $X_t = [U_t, z_t, c_t]$ is given by

$$\begin{aligned} c_t &= A_c c_{t-1} + \nu_{c,t},\ \nu_{c,t} \sim \mathcal{N}(0, \Sigma_c) \\ U_t &= g(U_{t-1}, z_{t-1}) \\ z_t &= (1 - c_t^T c_t)^{1/2} z_{t-1} + U_t vec^{-1}(c_t) \end{aligned} \tag{3.11}$$

The NSSA generative model is demonstrated in Fig. 3.8. The last equation follows from (3.4) and the fact that $v_t = U_t c'_t$, $U_t^* U_t = I$ and $c_t'^* c'_t = c_t^T c_t$. The above model is initialized with

$$z_0 = w_0,\ U_0 = left.singular.vectors(M(z_0)),\ c_0 = 0 \tag{3.12}$$

In the above model, we assume that $\nu_{c,t}$ is i.i.d Gaussian. We verified this fact by computing the histogram corresponding to each of the scalar dimensions of $\nu_{c,t}$ over the entire sequence (one of them is shown in Fig. 3.9) from a training sequence.

3.2.4 Model Parameter Estimation

The above model is completely specified by $z_{init} = w_0$, A_c, Σ_c. A_c is the AR transition matrix and Σ_c is the modeling error covariance matrix in the AR model. Given a training

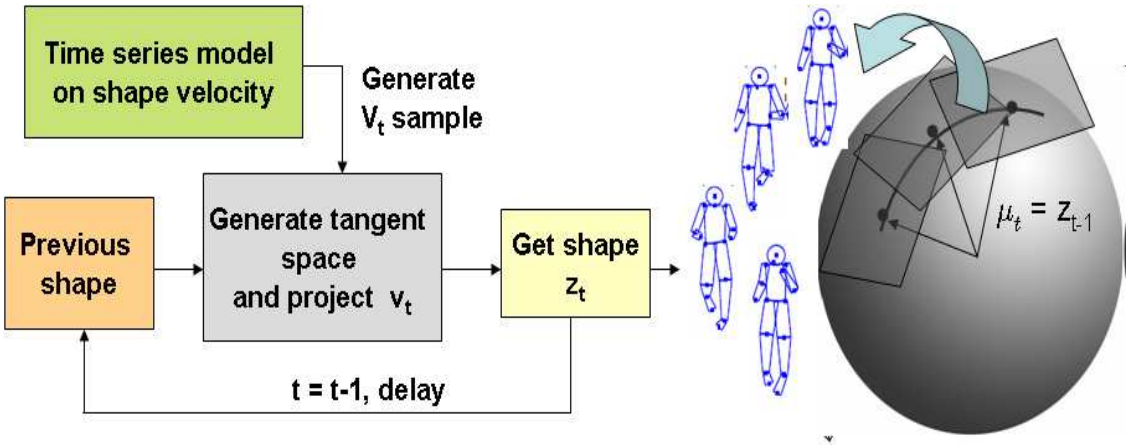


Figure 3.8 The NSSA generative model. Given the time series parameters for the evolution of shape velocities for a given activity we can generate the corresponding landmark shape sequence.

sequence of landmark configurations, $\{S_t\}_{t=0}^{N-1}$, a maximum-likelihood (ML) estimate of the parameters can be obtained as follows.

1. Obtain the shape sequence $\{z_t\}$ by translation, scale and rotation normalization of $\{S_t\}_{t=0}^{(N-1)}$ i.e. compute $y_t = C_K S_t$ and $w_t = \frac{y_t}{||y_t||}$ for each t. Set $z_0 = w_0$. Compute

$$z_t = w_t \frac{w_t^* z_{t-1}}{|w_t^* z_{t-1}|}, \ \forall t > 0 \tag{3.13}$$

2. For all t, obtain the shape velocity coefficients $\{c_t\}$ from $\{z_t\}$. This involves computing

$$\begin{aligned} U_t &= g(U_{t-1}, z_{t-1}) \\ c_t' &= U_t^* v_t = U_t^* z_t \\ c_t &= vec(c_t') \end{aligned} \tag{3.14}$$

starting with $U_0 = left.singular.vectors(M(z_0))$. The second equation above follows because $U_t^* v_t = U_t^*[I_K - z_{t-1} z_{t-1}^*] z_t = U_t^*[I_K - z_{t-1} z_{t-1}^* - 1_K 1_K^T/K] z_t = U_t^* U_t U_t^* z_t = U_t^* z_t$ (the second equality follows because z_t is translation normalized so that $1_K^T z_t = 0$ and third one follows because by definition, $U_t U_t^* = [I_K - z_{t-1} z_{t-1}^* - 1_K 1_K^T/K]$).

3. Obtain a maximum likelihood (ML) estimate of the AR model parameters, A_c, Σ_c, from

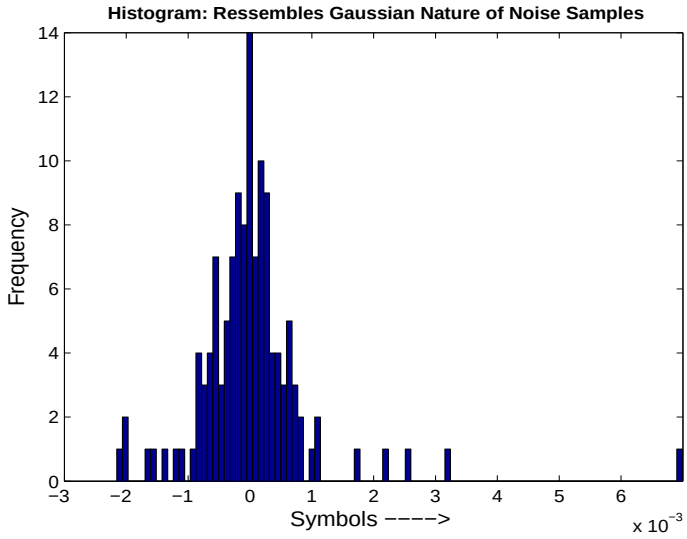


Figure 3.9 This figure shows the histogram corresponding to one of the scalar dimensions of $\nu_{c,t}$ over the entire sequence. Similar results were found for other dimension as well. These results validate the fact why we can consider samples of $\nu_{c,t}$ as i.i.d Gaussian.

$\{c_t\}$ by using the Yule-Walker equations, i.e.

$$\begin{aligned} A_c &= R_c(1)R_c(0)^{-1},\ where \\ R_c(0) &= \frac{1}{N}\sum_{t=0}^{N-1} c_t c_t^T,\ R_c(1) = \frac{1}{N-1}\sum_{t=1}^{N-1} c_t c_{t-1}^T \\ \Sigma_c &= \frac{1}{N-1}\sum_{t=1}^{N-1}(c_t - A_c c_{t-1})(c_t - A_c c_{t-1})^T \end{aligned} \tag{3.15}$$

3.2.4.1 Using Multiple Training Sequences

If more than one training sequence is available, one can compute a mean z_{init} (denoted by $\tilde{z}_{init}$) by aligning the initial preshapes, w_0, of all the sequences. We set z_0 for each sequence as the corresponding w_0 aligned to $\tilde{z}_{init}$. These operations make sure that the initial shapes of all the training sequences are aligned. Now, starting with z_0 we can obtain the the shape speed, c_t's for each sequence. Say, we have a total of q training sequences for a given motion activity, each with length N. We denote c_t's corresponding to the i^{th} sequence as $\{c_t^i\}$, where $i = 1, ..., q$. Now, we can estimate $R_c(0), R_c(1)$ as $R_c(0) = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{N}\sum_{t=0}^{N-1} c_t^i c_t^{iT}$ and $R_c(1) = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{N-1}\sum_{t=1}^{N-1} c_t^i c_{t-1}^{i\,T}$. Finally, we compute $A_c = R_c(1)R_c(0)^{-1}$ and $\Sigma_c = \frac{1}{q}\sum_{i=1}^{q}\Sigma_c^i$ where

Algorithm 4 2D NSSA: Training with Multiple Training Sequences For a Given Motion Activity

Input: Pre-shapes corresponding to q training sequences $(\{w_t^i\}_{t=0}^{N-1},\ i = 1, ..., q)$
Output: Computed parameters $\tilde{z}_{init}, A_c, \Sigma_c$.

1. Compute $\tilde{z}_{init} = \mu(w_0^1, w_0^2, ..., w_0^q,)$ where $\mu(.)$ is the Procrustes mean shape [28, 79].
2. Compute $\tilde{U}_{init} = left.singular.vectors(M(\tilde{z}_{init}))$ where M(.) is given in (3.8).
3. For each i , $i = 1, 2, \ldots q$
   (a) Compute $z_0^i = z(w_0^i, \tilde{z}_{init})$ using (3.2), compute $U_0^i = g(\tilde{U}_{init}, z_0^i)$ using (3.9) and set $c_0^i = 0$.
   (b) For each t, $t = 1, \ldots N-1$ do
      i. Compute $z_t^i,\ U_t^i,\ c_t^i$ using (13),(14).
4. Compute $A_c = R_c(1)R_c(0)^{-1}$ where,
   $R_c(0) = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{N}\sum_{t=0}^{N-1} c_t^i c_t^{iT}$ and
   $R_c(1) = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{N-1}\sum_{t=1}^{N-1} c_t^i c_{t-1}^{i\,T}$
5. Compute $\Sigma_c = \frac{1}{q}\frac{1}{N-1}\sum_{i=1}^{q}\sum_{t=1}^{N-1}(c_t^i - A_c c_{t-1}^i)(c_t^i - A_c c_{t-1}^i)^T$

$\Sigma_c^i = \frac{1}{N-1}\sum_{t=1}^{N-1}(c_t^i - A_c c_{t-1}^i)(c_t^i - A_c c_{t-1}^i)^T$. The entire procedure is summarized in Algorithm 4.

3.3 Modeling 3D Shape Sequences

A 3D configuration is represented by a set of K ordered landmarks as a $(K \times 3)$ matrix whose each row corresponds to the (x, y, z) coordinates of the corresponding landmark. In this section, we discuss the basics of 3D landmark shape analysis [28] and then develop 3D nonstationary shape activity model (3D-NSSA).

3.3.1 3D Landmark Shape Analysis Preliminaries

For 3D shapes, the computation of pre-shape $(w)_{K\times 3}$ from raw shape $(S)_{K\times 3}$ is similar to the 2D case i.e. first get the centered shape $(y)_{K\times 3}$ and then perform size normalization.

$$\begin{aligned} y &= C_K S \quad \text{where, } C_K \text{ is given in (3.2)} \\ w &= \frac{y}{||y||_F} \end{aligned} \tag{3.16}$$

Here, $||.||_F$ denotes *Frobenius norm* of a matrix. The rotation aligned shape z is obtained from pre-shape w in the following way. Say, we want to align $(w)_{K\times 3}$ w.r.t $(\mu)_{K\times 3}$. We do this as :

$$\begin{aligned} z &= w\mathcal{U}\mathcal{V}^T \quad \text{where,} \\ \mathcal{V}\Lambda\mathcal{U}^T &= SVD(\mu^T w) \end{aligned} \tag{3.17}$$

$\mathcal{V}$,$\mathcal{U}$ are the left and right singular vectors of the 3×3 matrix $(\mu^T w)$. It is to be noted that while performing 3D shape alignment we may have reflections unlike the 2D case. This happens if $det(\mathcal{U}) = -1$ or $det(\mathcal{V}) = -1$ where $det(.)$ denotes determinant. Thus 3D alignment is a bit different from 2D since reflections are allowed in 3D but not in 2D.

Another important thing about 3D shape analysis is the vectorization operation [28]. Say, $\mathbf{z}$ is the shape at a given instant which is a $(K\times 3)$ matrix with columns $\mathbf{z_1}, \mathbf{z_2}, \mathbf{z_3}$. We vectorize $\mathbf{z}$ to a $3K$ length vector as follows.

$$vec_{3D}(\mathbf{z}) = (\mathbf{z_1^T}, \mathbf{z_2^T}, \mathbf{z_3^T})^T \tag{3.18}$$

The inverse operation is given by $vec_{3D}^{-1}(.)$ which forms a $K\times 3$ matrix from a $3K$ length vector. The tangent space coordinate $v(z,\mu)$ of a shape z w.r.t the shape μ is given as follows,

$$v(z,\mu) = [I_{3K} - vec_{3D}(\mu)vec_{3D}(\mu)^T]vec(z) \tag{3.19}$$

The inverse map (i.e. from tangent space to shape space) is given as,

$$z = vec_{3D}^{-1}((1 - v^T v)^{\frac{1}{2}} vec_{3D}(\mu) + v) \tag{3.20}$$

3.3.2 3D Nonstationary Shape Activity (3D-NSSA)

To define an NSSA model on 3D shape data, we first obtain the translation and scale normalized pre-shape sequence $\{w_t\}$ from the 3D configuration sequence $\{S_t\}$ using (3.16). As in the 2D case, we use $\mu = z_{t-1}$ to compute the shape sequence followed by computing the shape velocity and shape speed vectors in an exactly analogous fashion. The final procedure can be summarized as follows.

First, we have $z_{init} = z_0 = w_0$ and then we compute the initial tangent space basis matrix as $U_{init} = U_0 = left.singular.vectors(M_{3D}(z_0))$ where,

$$M_{3D}(z) \triangleq [I_{3K} - vec(z)vec(z)^T]C_{K,3D} \tag{3.21}$$

where, $C_{K,3D} = \begin{bmatrix} C_K & \mathbf{0_{K\times K}} & \mathbf{0_{K\times K}} \\ \mathbf{0_{K\times K}} & C_K & \mathbf{0_{K\times K}} \\ \mathbf{0_{K\times K}} & \mathbf{0_{K\times K}} & C_K \end{bmatrix}$. Here, $\mathbf{0_{K\times K}}$ is a $(K \times K)$ matrix with all zero entries and C_K is defined in (3.2). Now starting with z_0 and U_0, the computation of the corresponding time sequence of *shape speed* vectors is done as follows,

$$z_t = w_t\mathcal{U}\mathcal{V}^T, \ where \ \mathcal{V}\Lambda\mathcal{U}^T = SVD(z_{t-1}^T w_t) \tag{3.22}$$

$$U_t = g(U_{t-1}, vec_{3D}(z_{t-1})) \tag{3.23}$$

$$c_t = U_t^T v_t = U_t^T vec_{3D}(z_t) \tag{3.24}$$

where, $g(.)$ is defined in (3.9). The reason why $U_t^T v_t = U_t^T vec_{3D}(z_t)$ is similar to the 2D case (see discussion below equation (3.14)).

We model c_t using a first order AR model (in general this may be replaced by any appropriate model for c_t). Thus, the forward model for generating a 3D shape sequence is:

$$\begin{aligned} c_t &= A_c c_{t-1} + n_t, \ n_t \sim \mathcal{N}(0, \Sigma_c) \\ U_t &= g(U_{t-1}, vec_{3D}(z_{t-1})) \\ z_t &= vec_{3D}^{-1}((1 - c_t^T c_t)^{\frac{1}{2}} vec_{3D}(z_{t-1}) + U_t c_t) \end{aligned} \tag{3.25}$$

The last equation follows from (3.20) and the fact that $v_t^T v_t = (U_t c_t)^T U_t c_t = c_t^T (U_t^T U_t) c_t = c_t^T c_t$ and $U_t^T U_t = I$.

3.3.3 Model Parameter Estimation

The parameter estimation algorithm for the 3D case can be summarized as follows.

1. For all t, obtain $\{w_t\}$ from a given 3D landmark configuration sequence, $\{S_t\}$.
2. For all t, compute $\{z_t\}$ from $\{w_t\}$ using (3.22).
3. For all t, compute $\{c_t\}$ from $\{z_t\}$ using (3.23) and (3.24).
4. Estimate the AR model parameters for c_t using the Yule-Walker equations given in (3.15).

3.4 Filtering and Tracking

The goal of filtering is to filter out the noise and get a good estimate of the true landmark shape from noisy observed landmarks. In our algorithm, the particle filter takes noisy observed landmark data as input and outputs the Minimum Mean Procrustes-distance Squared Error (MMPSE) estimate of the true landmark shape. The MMPSE estimate of shape can be derived by following the Procrustes mean derivation [28] as,

$$\begin{aligned}\hat{z}_t &= \arg\min_{\mu} E[d^2(z_t, \mu)|Y_{1:t}] \\ &= \arg\min_{\mu} E[||z_t z_t^* \mu - \mu||^2|Y_{1:t}] \\ &= \arg\max_{\mu} \mu^* E[z_t z_t^*|Y_{1:t}]\mu \end{aligned} \tag{3.26}$$

where $E[\cdot|Y_{1:t}]$ denotes the conditional expectation given $Y_{1:t}$, d denotes the Procrustes distance [28] and $Y_{1:t}$ are the observations until t. The last equality follows because $z_t^* z_t = 1$ and $\mu^*\mu = 1$. Under a particle filtering setup the MMPSE estimate is computed as, $\hat{z}_t$ = principal eigenvector of $\sum_{i=1}^{N_{pf}} z_t^i z_t^{i*} w_t^i$ where N_{pf} denotes number of particles, i denotes the i^{th} particle and w_t^i represents the importance weight corresponding to the i^{th} particle i.e. $p(z_t|Y_{1:t}) \approx \sum_{i=1}^{N_{pf}} w_t^i \delta(z_t - z_t^i)$. The configuration parameters i.e. scale, translation and rotation are also estimated in the process of filtering. Apart from removing random additive noise, particle filtering can also be used to 'clean up' the effects of occlusion and clutter.

Tracking is used to extract and filter out landmark configurations from a sequence of images. Filtering plays a very crucial role in the process. In fact, tracking can be considered as

observation extraction coupled with filtering. It works as follows. A shape deformation model (as described in Sec. 3.2.3) predicts the shape at the current instant using the previous shape estimates. Similarly, scale, translation and rotation models are used to predict their values as well. These, coupled with the predicted shape, gives the predicted landmark locations (i.e. predicted configuration) at the current instant. Using these predicted landmark locations in the current image, the landmarks can be extracted for the current image using any technique, for e.g. edge detection or optical flow. Our method for doing this is described in Algorithm. 7 and Sec. 3.4.5. Once the observed landmarks are obtained, they are filtered to get a MMPSE estimate of the true landmark shape and MMSE estimates of scale, rotation and translation. These estimates are again utilized to extract the observed landmark locations at the next time instant as described above.

We describe our state transition model and observation model in Sec. 3.4.1 and 4.6. We develop the PF algorithms for filtering in Sec. 3.4.3 and 3.4.4. The PF-based tracking algorithm to extract landmarks from video sequences is described in Sec. 3.4.5.

3.4.1 System Model (State Transition Model)

Since the observations are landmark configurations, to extract them, we need to estimate both the shape and the "motion" (scale, translation and rotation). Thus our state vector is, $X_t = [s_t, \theta_t, \tau_t, c_t, z_t, U_t]$ where s_t is the logarithm of global scale, θ_t is the global rotation, τ_t is the xy translation, c_t is the shape speed vector, z_t is the shape and U_t is the basis set spanning the current tangent space. The shape dynamics model is given in (3.11). It is a second order model on z_t which is equivalent to a first order model on the shape speed c_t. We use a first order model on logarithm of global scale, s_t, global 2D rotation θ_t (this typically models the random motion of camera) and translation τ_t.

$$\begin{aligned} s_t &= \alpha_s s_{t-1} + \nu_{s,t},\ \ \nu_{s,t} \sim \mathcal{N}(0, \sigma_s^2) \\ \theta_t &= \theta_{t-1} + \nu_{\theta,t},\ \ \nu_{\theta,t} \sim \mathcal{N}(0, \sigma_\theta^2) \\ \tau_t &= \tau_{t-1} + \nu_{\tau,t},\ \ \nu_{\tau,t} \sim \mathcal{N}(0, \sigma_\tau^2) \end{aligned} \tag{3.27}$$

Note that in case of filtering (when landmark observations are already available), translation

Algorithm 5 PF-Gordon for Landmark Shape Filtering

Initialization: At time $t = 0$, sample $s_0^{(i)} \sim \mathcal{N}(s_0, \sigma_s^2)$, $\theta_0^{(i)} \sim \mathcal{N}(\theta_0, \sigma_\theta^2)$, $c_0^{(i)} = \mathbf{0}$, $z_0^{(i)} = z_0$ and $U_0^{(i)} = U_0$. Here, $i = 1, ..., N_{pf}$ where, N_{pf} is the number of particles.
For $t > 0$,

1. Importance sample $X_t^{(i)} \sim p(X_t|X_{t-1}^{(i)})$ as $s_t^i \sim \mathcal{N}(\alpha_s s_{t-1}^i, \sigma_s^2)$, $\theta_t^i \sim \mathcal{N}(\alpha_\theta \theta_{t-1}^i, \sigma_\theta^2)$, $c_t^i \sim \mathcal{N}(A_c c_{t-1}^i, \Sigma_c)$, $U_t^i = g(U_{t-1}^i, z_{t-1}^i)$, $z_t^i = f(z_{t-1}^i, U_t^i, c_t^i)$. $i = 1, 2, ..., N_{pf}$
2. Weight and Resample. Compute $w_t^i = \frac{\tilde{w}_t^i}{\sum_{j=1}^{N_{pf}} \tilde{w}_t^j}$ where, $\tilde{w}_t^i = w_{t-1}^i p(Y_t|X_t^i)$ with, $p(Y_t|X_t^i) = p(Y_t|h(s_t^i, \theta_t^i, z_t^i)) = \prod_{k=1}^K [(1-p)\mathcal{N}([h(s_t^i, \theta_t^i, z_t^i)]_k, \sigma_o^2) + p\mathcal{N}(0, 100\sigma_o^2)]$, $\forall i$
3. Compute the MMPSE estimate $\hat{z}_t$ as the principal eigenvector of $\sum_{i=1}^{N_{pf}} z_t^i z_t^{i^*} w_t^i$. Estimate configuration parameters as $\hat{s}_t = \sum_{i=1}^{N_{pf}} w_t^i s_t^i$ and $\hat{\theta}_t = \sum_{i=1}^{N_{pf}} w_t^i \theta_t^i$
4. Set $t \leftarrow t + 1$ and go to step 1.

can be normalized for. Since it is a linear process, the form of the observation noise pdf does not change. But in case of tracking to predict and extract landmarks from image sequences, translation does need to be tracked to predict where the configuration of landmarks translated to.

The resulting state transition prior becomes,

$$\begin{aligned} p(X_t|X_{t-1}) = &\mathcal{N}(\alpha_s s_{t-1}, \sigma_s^2)\mathcal{N}(\theta_{t-1}, \sigma_\theta^2) \times \\ &\mathcal{N}(\tau_{t-1}, \sigma_\tau^2)\mathcal{N}(A_c c_{t-1}, \Sigma_c) \times \\ &\delta(U_t - g(U_{t-1}, z_{t-1}))\delta(z_t - f(z_{t-1}, U_t, c_t)) \end{aligned}$$

where δ denotes the Dirac delta function and $f(z_{t-1}, U_t, c_t) \triangleq (1 - c_t^T c_t)^{1/2} z_{t-1} + U_t vec^{-1}(c_t)$ and $g(.)$ is defined in (3.9).

3.4.2 Observation Model

There are various ways to extract landmarks from image sequences - e.g. edge detection followed by extracting the K strongest edges closest to predicted landmark locations or using the Kanade-Lucas-Tomasi (KLT) Feature Tracker [80] (block optical flow estimation) algorithm at or around predicted landmark locations. As explained in Sec. 3.4.5 and Algorithm. 7, we use a modification of KLT for this purpose.

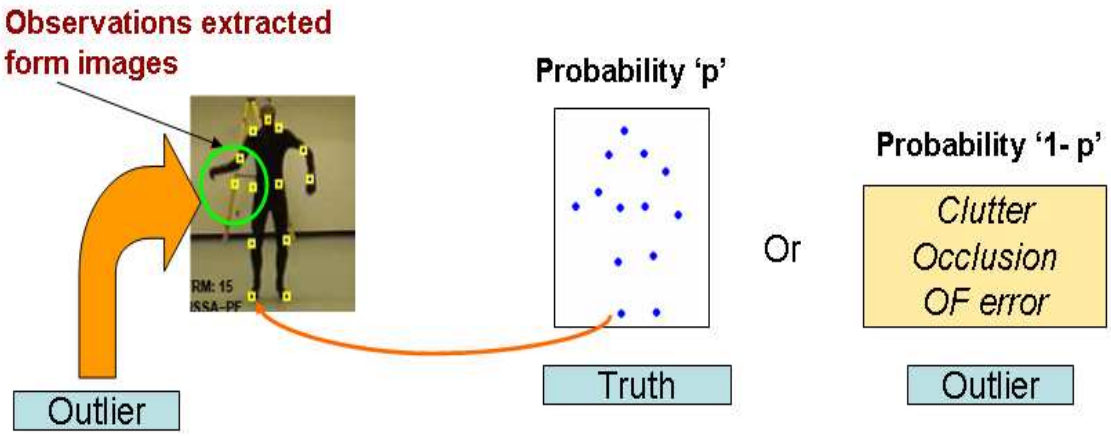


Figure 3.10 The observation mechanism is demonstrated here. Each observed landmark can either come from the true landmark location or generated from a outlier noise due to effects of occlusion and/or clutter.

The configuration of landmarks is obtained from the shape, scale and rotation by the transformation $h(s_t, \theta_t, z_t) = z_t e^{s_t} e^{j\theta_t}$. The simplest observation model is of the form

$$Y_t = h(s_t, \theta_t, z_t) + w_t, \;\; w_t \sim \mathcal{N}(0, \sigma_o^2 I) \tag{3.28}$$

where w_t is a complex Gaussian noise vector. This assumes that there is no background clutter: each of the K strongest edges or the K KLT-feature points are always generated by the true landmark location plus some error modeled as Gaussian noise. But this is often a simplistic model since there is always background clutter that generates false edges or false KLT-feature matches or there might be missing landmarks due to blur or occlusion. Thus it may happen that out of the K "observed landmark locations", some landmark at some time is actually generated by clutter (e.g. if a true landmark is blurred or occluded, while a nearby clutter point has a stronger edge). We model this as follows: with a small probability p, the k^{th} landmark, $Y_{t,k}$, is generated by a clutter point (model a clutter point location as a large variance Gaussian or by a uniform), independent of other landmarks. With probability $(1-p)$ it is generated by a Gaussian noise-corrupted actual landmark (independent of other landmarks), i.e.

$$Y_{t,k} \sim (1-p)\mathcal{N}([h(s_t, \theta_t, z_t)]_k, \sigma_o^2) + p\mathcal{N}(0, 100\sigma_o^2) \tag{3.29}$$

Algorithm 6 PF-EIS for Landmark Shape Filtering

Initialization: At time $t = 0$, sample $s_0^{(i)} \sim \mathcal{N}(s_0, \sigma_s^2)$, $\theta_0^{(i)} \sim \mathcal{N}(\theta_0, \sigma_\theta^2)$, $c_0^{(i)} = \mathbf{0}$, $z_0^{(i)} = z_0$ and $U_0^{(i)} = U_0$. Here, $i = 1, ..., N_{pf}$ where, N_{pf} is the number of particles.
For $t > 0$,

1. Importance sample $s_t^i \sim \mathcal{N}(\alpha_s s_{t-1}^i, \sigma_s^2)$, $\theta_t^i \sim \mathcal{N}(\alpha_\theta \theta_{t-1}^i, \sigma_\theta^2)$. $i = 1, 2, ..., N_{pf}$
2. Compute $m_t^i = \arg\min_{c_t} L^i(c_t)$ and $\Sigma_{IS}^i = [\nabla^2 L^i(m_t^i)]^{-1}$ where L^i is defined in (3.31).
3. Importance sample $c_t^i \sim \mathcal{N}(m_t^i, \Sigma_{IS}^i)$. Compute $U_t^i = g(U_{t-1}^i, z_{t-1}^i)$ and $z_t^i = f(z_{t-1}^i, U_t^i, c_t^i)$.
4. Compute Importance weights as, $w_t^i = \frac{\tilde{w}_t^i}{\sum_{j=1}^{N_{pf}} \tilde{w}_t^j}$ where, $\tilde{w}_t^i = w_{t-1}^i \frac{p(Y_t|h(s_t^i, \theta_t^i, z_t^i))\mathcal{N}(c_t^i; A_c c_{t-1}^i, \Sigma_c)}{\mathcal{N}(c_t^i; m_t^i, \Sigma_{IS}^i)}$.
5. Compute the MMPSE estimate $\hat{z}_t$ as the principal eigenvector of $\sum_{i=1}^{N_{pf}} z_t^i z_t^{i*} w_t^i$. Resample.
6. Set $t \leftarrow t + 1$ and go to step 1.

The above model has been adapted from the observation model used in Condensation [13]. The resulting observation likelihood term is,

$$p(Y_t|X_t) = \prod_{k=1}^{K} (1-p)\mathcal{N}([h(s_t, \theta_t, z_t)]_k, \sigma_o^2) + p\mathcal{N}(0, 100\sigma_o^2)$$

The intuitive idea behind the observation model is shown in Fig. 3.10.

3.4.3 Landmark Shape Tracking : PF with Efficient Importance Sampling

While keeping track of the landmark shapes using a particle filtering approach, we end up with large dimensional state space (number of landmarks could be a lot e.g. 10 or more). together with this, as shown in the observation model, we also have to deal with the effects of occlusion and clutter which gives rises to multimodality in the observation likelihood. Under these considerations and in light of our discussion in Chapter 2, we go for developing PF-EIS for landmark shape tracking problem.

3.4.4 PF-Gordon and PF-EIS for our problem

We summarize the basic PF (i.e. PF-Gordon [2]) for landmark shape filtering in Algorithm. 5. The idea of PF-Gordon for landmark shape tracking is demonstrated in Fig. 3.11. In order to develop the PF-EIS, we follow the steps as explained in Sec. 3.4.3. The choices of $X_{t,s}$ and $X_{t,r}$ under this problem set-up are justified as follows. Since the STP of s_t, θ_t is usually broad (to allow for occasional large camera motion or zoom), we use $X_{t,s} = [s_t, \theta_t]$

Algorithm 7 Automatic Landmark Extraction over a Sequence of Images

Input: image(t-1), image(t), $\hat{S}_{t-1}$ (estimated landmark configuration at $t-1$)
Output: $\{X_t^i, w_t^i\}, i = 1, 2, ..., \hat{S}_t = \sum_{i=1}^{N_{pf}} (z_t^i e^{s_t^i + j\theta_t^i} + \tau_t^i) w_t^i$ (estimated landmark configuration at time t) where, z_t is the shape and e^{s_t}, θ_t, T_t are the global scale, rotation and translation respectively.

1. For each estimated landmark $[\hat{S}_{t-1}]_k, k = 1, 2, \ldots K$, compute optical flow at a cluster of points around $[\hat{S}_{t-1}]_k$ and use this to move the points into image(t). Use the centroid of the moved cluster as the k^{th} observed landmark at time t, $[Y_t]_k$. Do this for all landmark points to get Y_t
2. Run PF-Gordon using Y_t, i.e. implement steps 1 and 2 of Algorithm. 5. But this time, we include the global translation in the state-space as well.
3. Display the estimated landmarks' location, $\hat{S}_t = \sum_{i=1}^{N_{pf}} (z_t^i e^{s_t^i + j\theta_t^i} + \tau_t^i) w_t^i$ (estimated landmark configuration at time t), set $t \leftarrow t+1$, and go to step 1.

and $X_{t,r} = [c_t, z_t, U_t]$. Note that for the purpose of importance sampling only s_t, θ_t, c_t are the "importance sampling states" since z_t, U_t are deterministically computed from c_t and X_{t-1}. The particles of $X_{t,s}$ are sampled from its state transition prior, i.e. using the first two equations of (3.27). Conditioned on the sampled scale and rotation, $X_{t,s}^i$, it is much more likely that p^* is unimodal, i.e. $p^{**,i}(c_t, U_t, z_t)$ defined below is unimodal

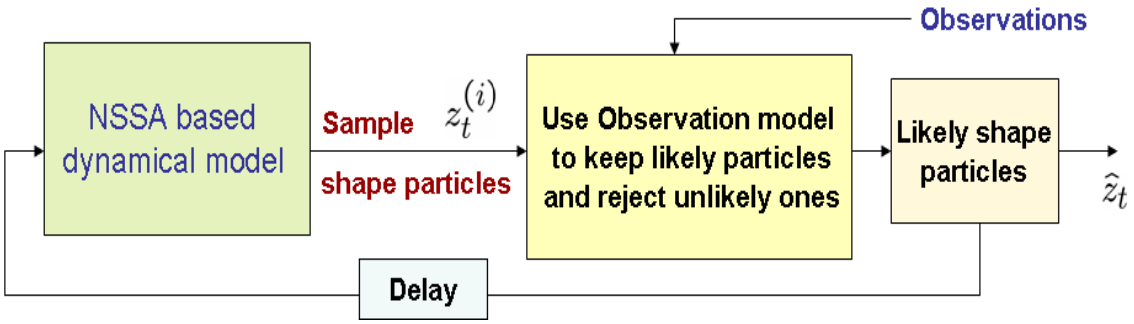


Figure 3.11 The basic particle filtering (i.e. PF-Gordon) idea for landmark shape tracking.

$$p^{**,i}(c_t, z_t, U_t) = \zeta\ p(\ Y_t \mid h(s_t^i, \theta_t^i, z_t)\ )\ \mathcal{N}(c_t; A_c c_{t-1}^i, \Sigma_c) \times$$
$$\delta(U_t - g(U_{t-1}^i, z_{t-1}^i))\delta(z_t - f(z_{t-1}^i, U_t, c_t))$$

where $\mathcal{N}(x; \mu, \Sigma)$ denotes the value of a Gaussian pdf with mean μ and variance Σ computed at the point x and ζ is a proportionality constant.

Since the pdfs of U_t, z_t, conditioned on c_t, X_{t-1}, are Dirac delta functions, the above simplifies to:

$$\begin{aligned} p^{**,i} &= \zeta \; p(\; Y_t \mid h(s_t^i, \theta_t^i, f(z_{t-1}^i, g^i, c_t)) \;) \; \mathcal{N}(c_t; A_c c_{t-1}^i, \Sigma_c) \times \\ &\quad \delta(U_t - g^i) \; \delta(z_t - f(z_{t-1}^i, g^i, c_t)) \\ &\triangleq p^{**,i}(c_t) \; \delta(U_t - g^i) \; \delta(z_t - f(z_{t-1}^i, g^i, c_t)) \end{aligned} \tag{3.30}$$

where, $g^i \triangleq g(U_{t-1}^i, z_{t-1}^i)$. The importance sampling part of $X_{t,r}$ is only c_t. We compute the importance density for c_t by approximating $p^{**,i}(c_t)$ by a Gaussian at its unique mode. The mode is computed by minimizing $L^i(c_t) = -\log p^{**,i}(c_t)$ defined below

$$\begin{aligned} L^i(c_t) &= [-\log p(\; Y_t \mid h(s_t^i, \theta_t^i, f(z_{t-1}^i, g^i, c_t)) \;)] + \\ &\quad [-\log \mathcal{N}(\; c_t; A_c c_{t-1}^i, \Sigma_c \;)] \end{aligned} \tag{3.31}$$

The PF-EIS algorithm for landmark shape tracking is summarized in Algorithm. 6.

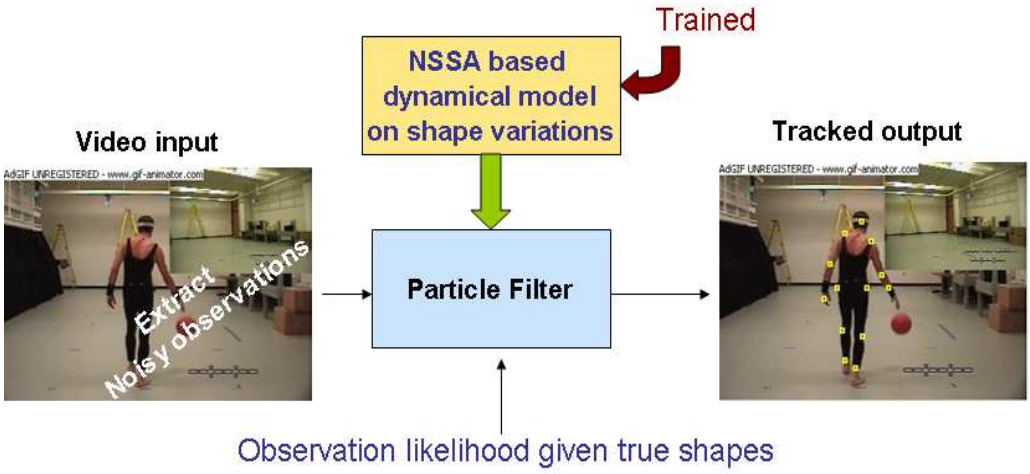


Figure 3.12 Automatic landmark extraction and tracking from videos using NSSA model.

3.4.5 Tracking to Automatically Extract Landmarks

In this section we describe out technique to track and automatically extract landmark configurations over a sequence of images or a video. The system comprises of a optical-flow (OF) tracker coupled with filtering. We compute optical flow at a cluster of points around each

currently estimated landmark location, $[\hat{S}_t]_k$ (k denotes k^{th} landmark) and use this to move the cluster of points into the next frame (frame $t+1$). The centroid of the moved cluster serves as the new observation for the k^{th} landmark at $t+1$. The same thing is done for all landmark points to get Y_{t+1} (the vector of observed landmark locations a $t+1$). This observation is fed into the NSSA-based PF which outputs the estimated landmark locations (and estimated shape) at $t+1$. The entire procedure is summarized in Algorithm. 7. For computing the optical flow we used the code/method developed by [81]. The idea is demonstrated in Fig. 3.12 and Fig. 3.13.

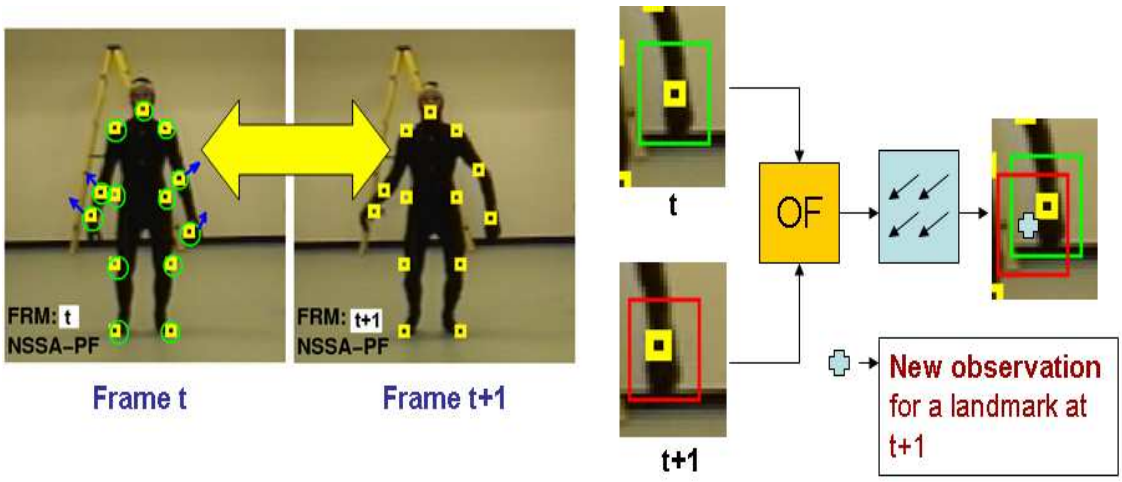


Figure 3.13 In this figure, the optical flow based observation extraction mechanism for automatic landmark extraction is demonstrated.

3.4.6 Change Detection / Abnormal Activity Detection in Video Sequences

In order to sense the change in shape activity model corresponding to an incoming video sequence, we use our NSSA model based automatic landmark extraction technique coupled with a Expected (negative) Log-Likelihood (ELL) based change detection statistics [40, 41].

An abnormal activity (suspicious behavior in our case) is defined as a change in the system model, which could be slow or drastic, and whose parameters are unknown. Given a test sequence of observations and a shape activity model, we use the change detection statistics defined in [40]. to detect a change (i.e. detect when observations stop following the given shape activity model). A change being drastic or slow depends on the system model used in

particle filtering. A more general system model can track a lot more changes and hence the nonstationary shape activity (NSSA) model is expected to do a better job of tracking abnormal observations than the stationary one. Whenever changed observations get tracked correctly, the ELL [40, 41] detects the change while if the PF loses track, the tracking error detects the change.

It is important to note that for abnormality detection, the normal activity needs to be characterized first. We can either use shape velocity or shape or both to represent normalcy depending on the practical problem being dealt with. To use shape for detecting abnormality, we would be required to represent a normal activity by a stationary shape activity model (SSA, [12]) or by a piecewise stationary shape activity (PSSA) model [41] (whichever is appropriate for a problem under consideration). We, however, choose to use shape velocity/shape speed statistics under our nonstationary system model setup.

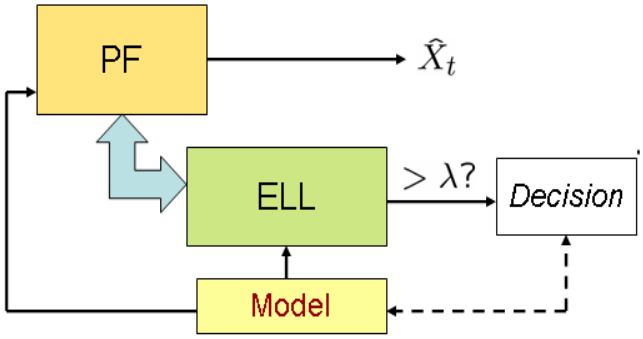


Figure 3.14 The integrated framework for tracking and change detection.

Since the NSSA model is a generic system model, it is expected to keep tracking (at least partially) even when the underlying motion activity changes. As long as the tracking error is under a certain threshold, the ELL statistics can give us a quantitative measure of how well/badly the observations are following the current system model. Under this assumptions, the ELL statistics on shape speed tells us how unlikely is the posterior estimate of the shape speed to have evolved from the prior dynamical model of the same. It is given as follows,

$$ELL(c_t) = E_{p(c_t|Y_{1:t})}(c_t^T \Sigma_c^{-1} c_t) \simeq \frac{1}{N} \sum_{i=1}^{N} {c_t^i}^T \Sigma_c^{-1} c_t^i \tag{3.32}$$

In the above equation, the c_t^i serve as the representative of the current motion activity given

the observations. Thus intuitively, the ELL statistics computes how far the current shape vectors deviate from their mean corresponding to the current prior model. Hence, if at time t, $ELL(c_t)$ has exceeded its threshold but the tracking error is still below its threshold (PF is still in track), we declare a change in activity. This can also serve as abnormal activity detection under the current problem setup. In case the PF looses track, the exploding tracking error can help us detect changes. It makes sense because PF performance is certain to go down as the underlying state sequence no longer follows the system model that the PF relies on. More details on change and abnormality detection can be found in [40, 8]. We use the automatic landmark extraction/tracking technique developed in our current research and validate the abnormality detection system for real-life video sequences. In the results section, we use this technique to detect a change in motion activity for run to jump on CMU Mocap data. The basic idea of integrating the change detection sytem with PF-based tracker is demonstrated in Fig. 3.14.

3.5 Model-based Compression of Landmark Shapes

In this section, we discuss the methodology developed for model based compression of 2D and 3D landmark shapes. We assume that translation and global scale are unimportant and need not be stored. We are only interested in storing the global rotation (in-plane rotation, especially for 2D shapes) and the shape change. Rotation is also not needed if it is due to camera motion, but some part of it may be due to actual rotation of the landmark configuration. *Thus our goal is to take the preshape sequence* $\{w_t\}$ *and compress it so as to achieve the minimum possible shape distortion for a given bit budget.*

3.5.1 2D NSSA model-based Compression

We began by first computing the differential entropy of the three possible models: NSSA, SSA and ASM, with model parameters learnt from the data. If the model assumptions are correct, the differential entropy is proportional to the entropy of the quantized data for small enough quantization size [82]. We realized that a measure of the average differential entropy

Algorithm 8 2D Landmark Shape Data Compression/Decompression

Required Inputs: The pre-shape sequence $\{w_t\}$, A_c at the transmitter side (compression) and $\{\tilde{n}_t\}$, $\{\tilde{\theta}_t\}$, z_{init} (i.e. z_0), A_c at the receiver side (decompression). Here, $\tilde{x}$ denotes the quantized version of x.

Initialize: $\tilde{z}_0 = w_0$, $\tilde{U}_0 = left.singular.vectors(M(z_0))$, $\tilde{c}_0 = c_0 = \mathbf{0}$. Computation of $M(z_0)$ is performed as mentioned in (3.8).

For $t > 0$,

(a) $z_t = w_t \frac{w_t^* \tilde{z}_{t-1}}{|w_t^* \tilde{z}_{t-1}|}$, $\theta_t = angle(w_t^* \tilde{z}_{t-1})$ Computing the aligned shape and the alignment angle

(b) $\tilde{U}_t = g(\tilde{U}_{t-1}, \tilde{z}_{t-1})$. Use equation (3.9) for this step. Performing basis alignment

(c) Compute $c_t = vec(\tilde{U}_t^* z_t)$ Computing shape speed vectors

(d) Compute $n_t = c_t - A_c \tilde{c}_{t-1}$ Computing prediction error under shape speed AR(1) model

(e) Quantize $\tilde{n}_t = Quantize(n_t)$, $\tilde{\theta}_t = Quantize(\theta_t)$. Transmit $[\tilde{n}_t, \tilde{\theta}_t]$

Decompressor (Implemented at each t at the TX end) :

(f) Compute $\tilde{c}_t = A_c \tilde{c}_{t-1} + \tilde{n}_t$ Reconstruct the quantized version of shape speed vectors

(g) Compute $\tilde{U}_t = g(\tilde{U}_{t-1}, \tilde{z}_{t-1})$

(g) Compute $\tilde{v}_t = \tilde{U}_t vec^{-1}(\tilde{c}_t)$, $\tilde{z}_t = (1 - \tilde{v}_t^* \tilde{v}_t)^{\frac{1}{2}} \tilde{z}_{t-1} + \tilde{v}_t$ Reconstruct the quantized shape

(h) Compute $\tilde{w}_t = \tilde{z}_t e^{-j\tilde{\theta}_t}$, $\tilde{w}_t^{RX} = \tilde{w}_t$ Reconstruc translation and size normalized configurations

end

Find $p_k(\alpha) = \frac{N(\tilde{n}_{t,k}=\alpha)}{N_{frames}}$, the PMF corresponding to the k^{th} dimension of $\{\tilde{n}_t\}$ for the alphabets α's. A Huffman table can thus be constructed for each scalar dimension. Entropy rate per scalar dimension is given as, $H_k = \sum_\alpha p_k(\alpha) \log_2(\frac{1}{p_k(\alpha)})$.

of NSSA was of the order of -250, which is much smaller than that of SSA(-200) or ASM(-195). This gives a preliminary indication that NSSA will indeed be a better model (if model assumptions are valid). Since a large part of the global rotation, θ_t, is often due to camera motion and hence independent of shape dynamics, we compress it separately from the shape sequence, z_t. Consider the NSSA model described in Sec. 3.2.3. The AR model prediction error, n_t, is assumed to be independent and identically distributed (i.i.d) Gaussian over time. We show in Fig. 3.9 that this assumption is a valid one for our datasets. Hence we propose to compute the sequence, $\{n_t\}$ (denoted as $\nu_{c,t}$ in Sec. 3.2.3), from the shape sequence, $\{z_t\}$, quantize it and store/transmit the Huffman coded version of quantized n_t.

Now, if the above quantization is done in an open-loop fashion - first compute the $\{n_t\}$ sequence and then quantize it, the reconstruction error at the decompression/receiver end will increase over time. This is because the quantization error in n_t will result in error in the

Algorithm 9 3D Landmark Shape Data Compression/Decompression

Required Inputs: The pre-shape sequence $\{w_t\}$, A_c at the transmitter side (compression) and $\{\tilde{n}_t\}$, $\{\tilde{\theta}_t\}$, z_{init}, A_c at the receiver side (decompression). Here, $\tilde{x}$ denotes the quantized version of x.

Initialize: $\tilde{z}_0 = w_0 = w_{init}$, $U_0 = left.singular.vectors(M_{3D}(z_{init}))$, $\tilde{c}_0 = c_0 = \mathbf{0}$. Computation of $M_{3D}(z)$ is given by the equation (3.21).

For $t > 0$,

(a) $z_t = w_t \mathcal{U}\mathcal{V}^T$ where, $\mathcal{V}\Lambda\mathcal{U}^T = SVD(\tilde{z}_{t-1}^T w_t)$ Computing the aligned shape w.r.t the previous one

(b) $\tilde{U}_t = g(\tilde{U}_{t-1}, vec_{3D}(z_{t-1}))$. Use equation (3.9) for this step. Performing basis alignment

(c) Compute $c_t = \tilde{U}_t^T vec_{3D}(z_t)$ Computing shape speed vectors

(d) Compute $n_t = c_t - A_c \tilde{c}_{t-1}$ Computing prediction error under shape speed AR(1) model

(e) Quantize $\tilde{n}_t = Quantize(n_t)$. Transmit $[\tilde{n}_t]$

Decompressor (Implemented at each t at the TX end) :

(f) Compute $\tilde{c}_t = A_c \tilde{c}_{t-1} + \tilde{n}_t$ Reconstruct the quantized version of shape speed vectors

(g) Compute $\tilde{U}_t = g(\tilde{U}_{t-1}, vec_{3D}(z_{t-1}))$

(g) Compute $\tilde{z}_t = vec_{3D}^{-1}((1 - \tilde{c}_t^T \tilde{c}_t)^{\frac{1}{2}} vec_{3D}(\tilde{z}_{t-1}) + \tilde{U}_t \tilde{c}_t)$ Reconstruct the quantized shape

(h) Define, $\tilde{z}_t^{RX} = \tilde{z}_t$

end

Find $p_k(\alpha) = \frac{N(\tilde{n}_{t,k}=\alpha)}{N_{frames}}$, the PMF corresponding to the k^{th} dimension of $\{\tilde{n}_t\}$ for the alphabets α's. A Huffman table can thus be constructed for each scalar dimension. Entropy rate per scalar dimension is given as, $H_k = \sum_\alpha p_k(\alpha) \log_2(\frac{1}{p_k(\alpha)})$.

estimate of c_t and hence of z_t, which in turn will propagate to the next time step - the error in z_{t+1} will be both due to error in n_{t+1} and due to the effect of errors in all past n_t's. This is a standard problem in all model-based compression schemes.

We use a standard solution to the above standard problem - we adopt a two-level Differential Pulse Coded Modulation (DPCM) scheme. Thus our encoding scheme involves implementing the receiver at the compression end itself before computing the next n_t, i.e. at each t:

1. Use the quantized version of n_t, denoted $\tilde{n}_t$, to compute $\tilde{c}_t = A_c \tilde{c}_{t-1} + \tilde{n}_t$

2. Compute the reconstructed shape $\tilde{z}_t$ using (3.4)

3. Compute $\tilde{U}_{t+1}$ which is the projection matrix for the tangent space perpendicular to $\tilde{z}_t$ (and close to $\tilde{U}_{t-1}$) using Gram-Schmidt given in (3.9).

4. Compute $c_{t+1} = vec(\tilde{U}_{t+1}^* z_{t+1})$ and $n_{t+1} = c_{t+1} - A_c \tilde{c}_t$ and quantize it.

The complete stepwise algorithm is summarized in Algorithm 8.

We use simple quantization to encode the rotation angle sequence, θ_t, although if a Markov model were assumed on θ_t then DPCM could be used there as well. We experimented with both approaches, with negligible difference in performance.

3.5.2 3D NSSA model-based Compression

The 3D landmark shape sequence compression algorithm is very similar to the 2D case discussed above. Similar to the 2D case, we also use a two-level Differential Pulse Coded Modulation (DPCM) scheme to avoid reconstruction error propagation. In fact, once we have the shape velocity coefficients (using either 2D or 3D NSSA) the compression schemes are the same for 2D and 3D. We just have to make sure that we reconstruct the shapes and perform the shape alignments using appropriate methodologies (as described in Sec. 3.3). In the 3D landmark case, we reconstruct the aligned shape sequence only ($\{z_t\}$) and not the preshapes i.e. we ignore the rotation information. Hence, we are not required to encode/store the rotation parameter or the information about $\mathcal{U}\mathcal{V}^T$ for each frame (see equation (3.22)). The complete stepwise algorithm for 3D NSSA based compression is summarized in Algorithm 9.

3.6 Performance Evaluation and Comparison

In this section, we develop performance evaluation matric for the compression schemes described earlier. We then compare the compression vs. distortion performances of NSSA with other methods.

3.6.1 Performance Evaluation Metric

To compare the performance of NSSA-based DPCM, we applied an exactly analogous scheme to the SSA model and to the ASM model. We varied the number of quantization bits per unit time per scalar dimension from 4 to 10 and plotted the mean squared distortion of the preshape against the Huffman-encoded bit rate (R-D plot) per unit time. The Huffman-coded

bit rate will always be within one bit of the entropy rate [82]. For the 2D case it is defined by,

$$H(b) = \sum_{k=1}^{M} H_b(\tilde{n}_{t,k}) + H_b(\theta) \tag{3.33}$$

Where, $H_b(\tilde{n}_{t,k})$ is the entropy rate for the k^{th} dimension of $\tilde{n}_t$, given the word-length $\mathbf{b}$. M is the dimensionality of n_t and $H_b(\theta)$ is the corresponding entropy rate for $\tilde{\theta}_t$. For the 3D case, however, we do not add the entropy corresponding to the rotation parameter (unlike $H_b(\theta)$ in the 2D case). $H_b(\tilde{n}_{t,k})$ is computed as,

$$H_b(\tilde{n}_{t,k}) = \sum_{\alpha} p_k(\alpha) \log_2(\frac{1}{p_k(\alpha)}) \tag{3.34}$$

Where, $p_k(\alpha) = \frac{N(\tilde{n}_{t,k}=\alpha)}{N_{frames}}$, the PMF corresponding to the k^{th} dimension of $\{\tilde{n}_t\}$ for the alphabets α's.

The mean squared distortion for the 2D case is defined as,

$$D = \frac{1}{N_{time}} \sum_{t=1}^{N_{time}} ||w_t - \tilde{w}_t^{RX}||^2 \tag{3.35}$$

Where, w_t is the original preshape at the transmitter and $\tilde{w}_t^{RX}$ is the reconstructed preshape at time instant t. For the 3D case, we compute the distortion as the error measure between the true aligned shape z_t and it's reconstructed version at the receiver side i.e. $\tilde{z}_t^{RX}$. We compute the distortion per unit time and entropy rate (or Huffman-coded bit rate) for each video sequence and plot their average values over all sequences (a total of 80). The corresponding results will be discussed in the section to follow.

Note that, in all schemes (NSSA, SSA, ASM), one also needs to accurately quantize and store the Huffman table, the AR matrix, A_c, the initial shape z_{init} (or the mean shape μ in case of SSA or ASM). These will require the same number of bits for all methods and hence are not compared. Also, this will be a one-time cost and its effect on the total number of bits will become negligible as the length of the sequence increases.

3.7 Experimental Results

We began by comparing the ability of NSSA to model real-life landmark shape sequences with that of SSA, ASM and the wrong NSSA model (NSSA-unaligned) [41]. This is discussed

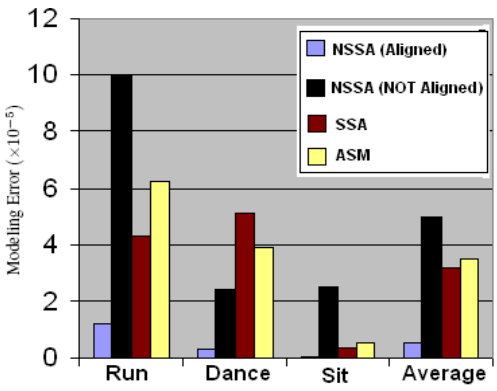
in Sec. 3.7.1. It was observed that NSSA had much smaller modeling error than all three. This comparison gave us the first indication that NSSA would provide a much more accurate prior dynamic model for Bayesian filtering or tracking applications, as well as also for synthesis/extrapolation applications.

Next we simulated multiple realizations of a "nonstationary shape activity" along with scale and rotation variations and attempted to track it using three different PF algorithms: the original PF (PF-Gordon) was compared with PF-Doucet [3] and PF-EIS [15] (described in Sec. 3.4.3). These comparisons are discussed in Sec. 3.7.2. It was observed that when the number of available particles is small, PF-EIS has the best performance.

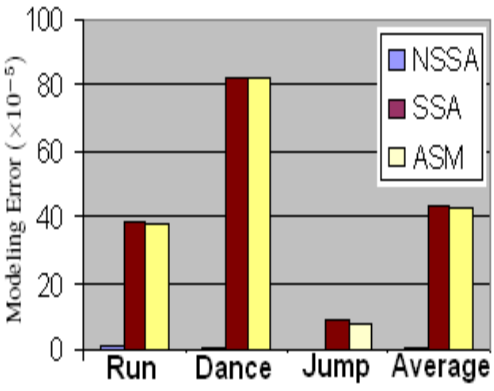
Since most existing work that used SSA or ASM for tracking used PF-Gordon, we retained this PF for most of our comparisons between NSSA, SSA and ASM. The following four sets of experiments were done. In Sec. 3.7.3, we demonstrate the superior ability of NSSA-based PF (PF-Gordon with $N_{pf} = 1000$ and PF-EIS with $N_{pf} = 50$) to filter out the landmark shapes from heavily noise-corrupted and cluttered observed landmark configurations. In Sec. 3.7.4, we compare the tracking ability of NSSA-based PF with SSA-based PF and ASM-based PF, i.e. their ability to accurately extract out landmarks from image sequences. Once again NSSA is found to be significantly superior to SSA and ASM and also to direct landmark extraction (without any model-based filtering). The use of 3D-NSSA to accurately synthesize new human activity sequences is discussed in Sec. 3.7.5. In Sec. 3.7.6 we give a preliminary experiment that demonstrates that NSSA is able to remain in track even when a model change occurs. The performance evaluation results for 2D/3D landmarks' shape data compression are given in Sec. 3.7.7.

3.7.1 Modeling Error Comparison

We used the Carnegie Mellon Motion Capture (CMU Mocap) database [10] for our experiments. Each file in the database had the coordinates of the markers placed at the body landmark locations (especially the body joints) for successive frames of a specific human motion activity e.g. running, jumping etc. An example is shown in Fig. 3.21. The corresponding



(a) Modeling error for 2D shape data



(b) Modeling error 3D shape data

Figure 3.15 Fig. 3.15(a) shows the modeling error (ME) for NSSA, ASM and ASM for a few activities using 2D MOCAP data. It is important to note that NSSA, without the basis alignment has a very large modeling error. While after the basis alignment is taken into account, NSSA has much lower ME than SSA and ASM. In Fig. 3.15(b) we show modeling error(ME) for NSSA, ASM and ASM for a few activities using 3D MOCAP data. Again NSSA had much lower ME compared to that SSA and ASM. That is why the corresponding bar plot for NSSA modeling error has almost disappeared.

2D and 3D landmark shape sequences were used as the training data for learning the NSSA (or SSA or ASM) model parameters.

We define modeling error (ME) as the trace of the noise covariance matrix of the AR modeling error, i.e. ME $= trace(\Sigma_c)$, where $\Sigma_c = E[(c_t - A_c c_{t-1})(c_t - A_c c_{t-1})^T]$ and c_t are the "aligned" coefficients of shape velocity. We also compute the modeling error when the c_t's are not aligned, i.e. when U_t is computing using SVD at each t (as in [41]). When computing error for SSA, c_t are the tangent space coefficients of shape (not of shape velocity), i.e. all shapes z_t are projected into a single tangent space at the common mean shape μ. For ASM, modeling error is still the same but now $c_t = z_t - \mu$, i.e. the shape space is assumed to be Euclidean.

We computed the modeling error of SSA, ASM and NSSA (unaligned) for various human actions and compared with that of NSSA. It was found that NSSA has much lower modeling error than all three. The modeling error bar plots of 2D and 3D shape sequences have been shown in Fig. 3.15(a), 3.15(b) for a few motion activities.

Modeling error tells us how well we are able to capture the shape deformation dynamics using the given model. It thus quantifies how well we can predict the shape at a time instant

given the information from the previous time instant. Thus Lower modeling error will result in better tracking ability and also a better ability to synthesize a new sequence or to extrapolate an existing sequence (graphics problems).

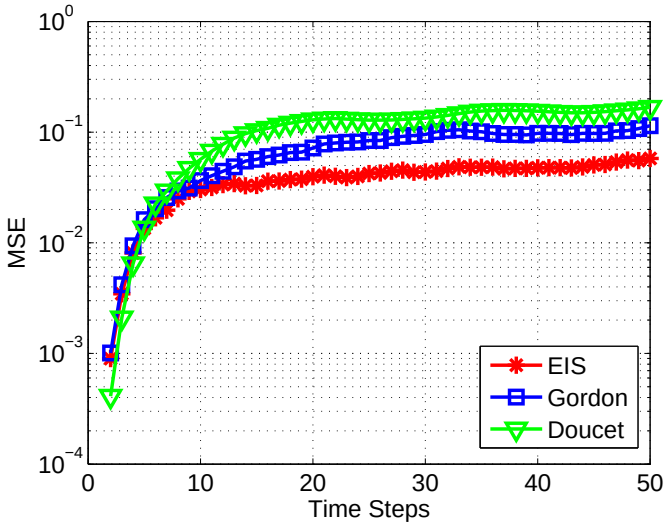


Figure 3.16 In this Figure, we compare the MSE of of estimated configurations computed using PF-EIS, PF-Doucet and PF-Gordon for simulated shape sequence. EIS has the smallest MSE (discussed in Sec. 3.7.2).

3.7.2 Comparing PF algorithms

We simulated landmark shape change of a set of $K = 5$ landmarks (a deforming pentagon) and tracked it using PF-EIS (Algorithm. 6), PF-Gordon (Algorithm. 5) [2] and PF-Doucet [3] with $N_{pf} = 50$ particles. The initial shape, z_0, was a regular pentagon. The shape and global motion change of the configuration followed (3.11), (3.27) with $\Sigma_c = 0.0025 I_6$, $\sigma_s^2 = 0.0001$, $\sigma_\theta^2 = 0.25$, $A_c = 0.6 I_6$, $\alpha_s = 0.9$. The observations followed (3.29) with $\sigma_o^2 = 0.04$ and $p = 0.2$. I_6 denotes a 6×6 identity matrix. It is to be noted that the state transition prior (STP) of scale (e^{s_t}) is a log-normal and hence even $\sigma_s^2 = 0.0001$ results in a fairly broad distribution.

Whenever one or more landmarks are generated by clutter, the observation likelihood (OL) of log-scale (s_t) is either heavy-tailed with the wrong (outlier) mode or is multimodal. When many landmarks are generated by clutter, the same happens for θ_t. This combined with a broad prior of s_t, θ_t, results in multimodal $p^*(s_t, \theta_t)$. Whenever this happens, most particles of

PF-Doucet end up sampling from a Gaussian about the wrong mode of $p^*(s_t, \theta_t)$ or of $p^*(s_t)$, resulting in loss of track. But PF-EIS does not suffer from this problem since it samples from the prior of s_t, θ_t. Also, since the prior of c_t is narrow compared to the distance between likelihood modes, $p^{**,i}(c_t)$ is usually unimodal and thus sampling from its Laplace's approximation is valid. On the other hand, for small N_{pf}, PF-Gordon often loses track because it samples all states from the prior, thus resulting in small effective particle size. In Fig. 3.16, the mean squared error (MSE) plot averaged over 80 Monte Carlo runs is shown. Also see Fig. 3.17 for MOCAP data, where PF-EIS with $N_{pf} = 50$ particles is able to achieve almost the same tracking accuracy as PF-Gordon with $N_{pf} = 1000$.

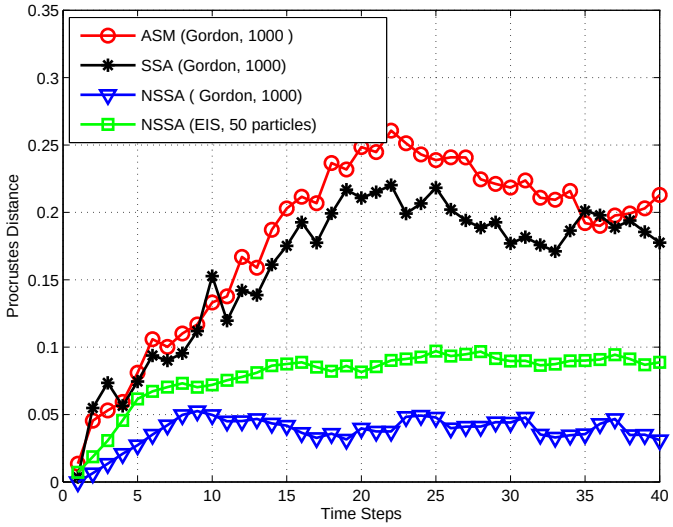


Figure 3.17 Filtering noise and clutter corrupted MOCAP data: comparing NSSA, SSA and ASM based PF-Gordon and also NSSA based PF-EIS. NSSA significantly outperforms SSA and ASM. NSSA-based PF-EIS with just 50 particles has comparable performance to that of 1000 particle NSSA-based PF-Gordon (discussed in Sec. 3.7.3).

3.7.3 Filtering from Noisy and Cluttered Observations

In this experiment, 2D landmark shape sequences from CMU Mocap database (refer Sec. 3.7.1) were used as the ground truth (shown in Fig. 3.19, top row). We incorporated random scale variations, additive noise and clutter to the ground truth. The observations were simulated using (3.29). This experiment helped us to quantitatively compare performance since

ground truth was available. The filtering mechanism is demonstrated in Fig. 3.18.

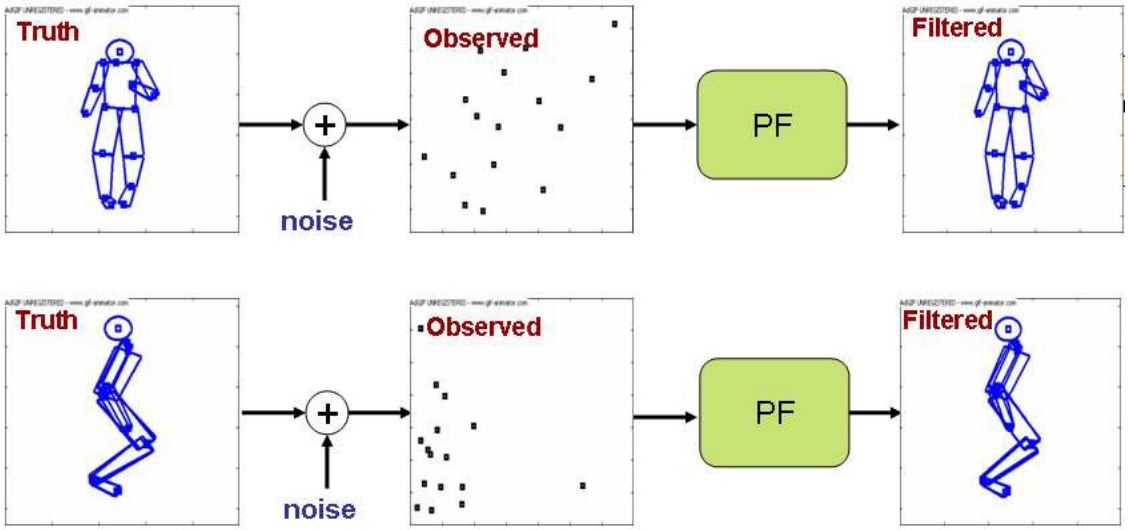


Figure 3.18 Sequential filtering of true landmark shapes out of noise observed set of landmarks.

We used the PF-Gordon (i.e. the basic PF) with $N_{pf} = 1000$ particles for filtering. Our primary objective was to compare the performance of NSSA based system model with that of SSA and ASM. PF-Gordon solely depends on the state transition prior (STP) for importance sampling and thus its performance heavily relies on the accuracy of the system model. Also, all past work on SSA or ASM based tracking used PF-Gordon.

We considered two motion activities namely *run* and *jump* with $K = 16$ landmarks. Different data sequences were used for training and testing. Our results are shown in Fig. 3.19 (run) and Fig. 3.20 (jump) and the procrustes distance comparison plots are given in Fig. 3.17. The landmark locations were fitted with rectangular patches to visualize the body posture at a given instant. The first row shows the ground truth and also the observation. It can be seen that the observed landmark locations (represented as $+$ on top of the ground truth) were severely corrupted with clutter and random noise. Thus, visually, the observed configurations hardly conform to the human body. But, as shown in the second row of Fig. 3.19, NSSA has been able to filter out the true shape from those observations with very good accuracy. SSA (third row) and ASM (fourth row), however, perform poorly in this job. Similar results were

found for motion activity *jump* (see Fig. 3.20). In Fig. 3.17, we plot the procrustes distance between the estimated shape and the true shape at each instant as the quantitative measure of filtering accuracy. Also note that PF-EIS (with NSSA model) is able to achieve almost the same accuracy as PF-Gordon (with NSSA model) with as few as $N_{pf} = 50$ particles.

In case of SSA and ASM, filtering performed around the fixed mean shape ends up generating shape samples far away from the desired shape space where the original set of deforming shapes lie. This, in turn, led to their poor performances. But NSSA, on the other hand, does an excellent job in filtering because of its time varying mean shape assumption. Of course, we assume that the shape deviations over successive time frames are small enough to make sure the the current shape is in the neighborhood of the current mean shape (i.e. the previous shape) for our mathematical treatments to be valid. This is a reasonable assumption for most of the real-life motion activities.

3.7.4 Tracking and Automatic Landmark Extraction

For automatic landmark extraction and tracking, we used 50 frames of real-life videos of human activities (shown in Fig. 3.21(run) and Fig. 3.22(jump)). A total of 13 body landmarks were considered as shown in Fig. 3.22. The body landmarks were: right shoulder, right elbow, right hand, right hip, right knee, right foot, head, left shoulder, left elbow, left hand, left hip, left knee, left foot.

For run sequences, the system model was learnt from the hand-marked ground truth data corresponding to the training database. In case of jump, however, we used the mocap data itself. We appropriately subsampled the mocap data frames in order to roughly synchronize them to the video frames. Implementations of Algorithm. 7 using system models based on based on NSSA, SSA and ASM were compared. Their performances over multiple time frames are shown in Fig. 3.21 (run). It can be clearly seen that NSSA (top row) performs much better than SSA (second row) or ASM (third row) in terms of *keeping track* of the body landmarks. It is very important to note that filtering and prediction play a very crucial role in the process of extracting landmark observations. To verify this, we extracted the landmark observations

purely based on optical flow i.e. starting with the initial locations we used the optical flow between successive frames to drift the points and thus getting observations over time. This procedure does not use the knowledge of the state estimates at each instant to decide where the expected landmark might be while computing the optical flow. As can be seen from Fig. 3.21 (fourth row), quite a few of the observed landmarks in this case were found to get stuck at wrong locations (due to background interference, occlusion or clutter) and from that point on they were never in sync with the body movements. In fact, the point of using the state estimates is to correct the locations of such landmarks and making sure that we do not end up computing the OF at the wrong regions for getting the landmark location for the next frame.

Next, we test the system with NSSA based system model on the video of a person *jumping*. It can be seen in Fig. 3.22 that NSSA did a very good job in terms of tracking the landmarks over various time frames. It is to be noticed that at frame 15, it loses track of the landmark corresponding to the right hand. But later, it regains the track (see frame 39, Fig. 3.22).

3.7.5 3D-NSSA Synthesis

Motivated by the drastically small modeling error of 3D-NSSA for human actions (see Fig. 3.15(b)), we attempted to use 3D NSSA for a motion synthesis application. We learnt the deformation model for 3D body shape of the human body while running from MOCAP data. The synthesized run sequence using this model is shown in Fig. 3.23. Since there is no ground truth is this case, we visually verified the systems ability to synthesize *run* and the system performance was found to be promising.

3.7.6 Change Detection with NSSA

We did a simple experiment to test the ability of the NSSA-based tracker to detect changes in activity while still not completely losing track. We used a sequence where a person begins by running and then leaps (`http://mocap.cs.cmu.edu:8080/subjects/49/49_04.avi`). Notice that this is a fairly sudden change. The ELL-based change detection statistic [40] was able to detect the change to leap, and for sometime after the change also, our tracker did not

lose track (see Fig. 3.24 - tracking error does not increase until later). More results and comparison with SSA are shown in [41]. Detailed results and all MATLAB codes can be found at `http://www.public.iastate.edu/~samarjit/pami.html`.

3.7.7 Experimental Results : Landmark Shape Compression

We compared the performance of NSSA model-based compression with that of stationary shape activity (SSA) and active shape model (ASM) and nonstationary ASM (NS-ASM). The SSA [8, 12] model used a fixed mean shape μ and then defined an AR or ARMA model on the computed c_t, i.e. z_t was assumed to be stationary. In our comparisons, we use an AR model on c_t. Thus, SSA computed n_t, θ_t as follows

$$\begin{aligned} c_t &= vec(U(\mu)^*[I - \mu\mu^*]z_t) = vec(U(\mu)^* z_t) \\ n_t &= c_t - A_c c_{t-1} \\ \theta_t &= angle(w_t^* \mu) \end{aligned} \tag{3.36}$$

and then compressed them using an algorithm analogous to Algorithm. 8. Here $U(\mu)$ denotes the orthonormal basis spanning the tangent space at the fixed mean shape μ. On the other hand, ASM (or PDM)[29] assumed both stationarity and the fact that z_t belongs to a Euclidean space. In ASM, the first equation of (3.36) is replaced by $c_t = vec(z_t - \mu)$ and everything else is the same as that for SSA.

We also compare with the case when we assume that the shape space is Euclidean, but define a second order model on z_t similar to that of NSSA. We call this "nonstationary ASM" (NS-ASM). In this case n_t is computed as

$$\begin{aligned} c_t &= vec(z_t - z_{t-1}) \\ n_t &= c_t - A_c c_{t-1} \end{aligned} \tag{3.37}$$

3.7.8 2D landmark shape data compression

We began by first comparing the differential entropy [82] of the NSSA model with that of the other models. In each case, the model parameters were learnt for the same dataset.

If the model assumptions are correct, the differential entropy is proportional to the entropy of the quantized data for small enough quantization size [82]. We found that a measure of the differential entropy averaged over various motion activities corresponding to NSSA was -250, which was smaller than that of SSA(-200) or ASM(-195) or NS-ASM(-215). This gives a preliminary indication that NSSA will indeed be a better model for compression (if model assumptions are valid).

To compare the performance of NSSA-based DPCM with that of existing work, we used a DPCM based compression algorithm analogous to Algorithm. 8 but using the SSA, ASM or NS-ASM model instead of the NSSA model. Since the goal is to compare the different models, we only performed scalar quantization followed by Huffman coding. We varied the number of quantization bits per unit time per scalar dimension from 4 to 10 and plotted the mean squared distortion of the preshape against the Huffman-encoded bit rate per unit time. This is commonly referred to as the Rate-Distortion (RD) plot. The Huffman-coded bit rate will always be within one bit of the entropy rate of $\tilde{n}_{t,k}$ [82] which is computed as

$$H(b) = \sum_{k=1}^{M} H_b(\tilde{n}_{t,k}) + H_b(\tilde{\theta}_t) \tag{3.38}$$

where $H_b(\tilde{n}_{t,k})$ is the entropy rate for the k^{th} dimension of $\tilde{n}_t$, given the word-length b. M is the dimensionality of n_t and $H_b(\tilde{\theta}_t)$ is the corresponding entropy rate for $\tilde{\theta}_t$. Our shape sequence had 16 landmarks and so $M = 2 * 16 - 4 = 28$. The entropy $H_b(\tilde{n}_{t,k})$ is computed as,

$$H_b(\tilde{n}_{t,k}) = \sum_{\alpha} p_k(\alpha) \log_2(\frac{1}{p_k(\alpha)}) \tag{3.39}$$

where $p_k(\alpha) = \frac{N(\tilde{n}_{t,k}=\alpha)}{N_{frames}}$, the probability mass function (PMF) corresponding to the k^{th} dimension of $\{\tilde{n}_t\}$ for the alphabets α's. Here $N(\tilde{n}_{t,k} = \alpha)$ denotes the number of times $\tilde{n}_{t,k}$ is equal to α for various t.

The mean squared distortion is defined as,

$$D = \frac{1}{N_{time}} \sum_{t=1}^{N_{time}} ||w_t - \tilde{w}_t^{RX}||^2 \tag{3.40}$$

where w_t is the original preshape at the transmitter and $\tilde{w}_t^{RX}$ is the reconstructed preshape at time instant t (see step(h) of Algorithm. 8).

We used motion capture data from the CMU MOCAP database for our comparisons. The motion activity sequences used in these experiments were - run, jump, dance etc. A total of 80 different sequences were used. In the encoding process, we tried seven different quantization word-lengths ranging from 4 to 10 bits per unit time per scalar dimension. For each word-length, we computed the Huffman coded bit rate (which is within a bit of the average entropy rate per frame) and the average distortion over all the video sequences. Thus we got the rate-distortion (R-D) plot of the system. We compare the compression performance of NSSA, SSA, ASM and NS-ASM using their corresponding RD curves in Fig. 3.25 (left). For similar Huffman-encoded bit rates, the NSSA based method gave a distortion of the order of 10^{-5}, while distortions corresponding to SSA and ASM were of the order of 10^{-3} and 10^{-2} respectively and that of NS-ASM was 10^{-4}. A visual comparison between NSSA and ASM decompression has been shown in Fig. 3.26.

Another measure of the system performance for a specific word-length, b, is the peak signal-to-noise ratio (PSNR) defined as,

$$PSNR(b) = 10\log_{10}(\frac{||w_t||^2}{D_{avg}(b)}) = 10\log_{10}(\frac{1}{D_{avg}(b)})$$

With $||w_t|| = 1$ by definition, PSNR values for $b = 4$ were 30 dB for NSSA, 23.5 dB for SSA and 17.2 dB for ASM respectively.

It is to be noted that, in all schemes (NSSA, SSA, ASM, NS-ASM), one also needs to accurately quantize and store the Huffman table, the AR matrix, A_c, the initial shape z_{init} (or the mean shape μ in case of SSA or ASM). These will require the same number of bits for all methods and hence are not compared. Also, this will be a one-time cost and its effect on the total number of bits will become negligible as the length of the sequence increases.

3.7.9 3D landmark shape data compression

For the 3D case, as explained earlier, we only encode the shape sequence and not the rotation information. We compute the distortion as the error measure between the true aligned shape z_t and its reconstructed version at the receiver side i.e. $\tilde{z}_t^{RX}$. Distortion is computed as the Frobenius norm of $[z_t - \tilde{z}_t^{RX}]$. We compute the distortion per unit time and the Huffman-

coded bit rate for each video sequence and plot their average values over all sequences (a total of 80) in Fig. 3.25 (right). The distortion using 3D NSSA is of the order of 10^{-8} to 10^{-11} which is much smaller than that of SSA (of the order of 10^{-6}), ASM (of the order of 10^{-5}) and NS-ASM (about 10^{-7}).

3.7.10 Shortcomings of NSSA: Classification

Despite very good performances while tracking/filtering, in its current form, NSSA does not perform as well for classification. We tried to perform a model based maximum likelihood classification among various motion activities (eg. run, jump, sit etc.). The input to the classifier was the time sequence of shape velocity coefficients for NSSA, tangent space coefficients for SSA, and shape deviation vectors for ASM. The output was the most likely activity to have generated the sequence. In our preliminary experiments with run, sit, jump and dance activity sequences, NSSA had a 4% misclassification rate while SSA and ASM had 2.5% and 2% respectively. The reason NSSA does not perform as well as the rest is the same as the reason it significantly outperforms SSA and ASM for tracking - it is a more generic model for shape change. It consists of a zero mean random walk model on shape and a zero mean AR model on shape velocities. The effect of initial shape is lost in a long sequence.

To use NSSA for classification, we should modify the current model and define a nonzero-mean shape velocity change model. Alternatively, a good idea would be to use NSSA for tracking, i.e. for extracting landmark shape sequences from video, and then feeding these into a piecewise SSA [32] or piecewise ASM [31] based classifier.

3.8 Summary

The key contribution of this work is a novel approach to define a generative model for both 2D and 3D nonstationary landmark shape sequences, which we call nonstationary shape activity (NSSA). The main idea is to compute the tangent space representation of the current shape in the tangent space at the previous shape. This can be referred to as the shape velocity vector since it quantifies the difference between two consecutive shapes projected into

the tangent space at the first one. The "aligned" shape velocity coefficients (shape speed vector) are modeled using standard vector time series methods. Applications in filtering, tracking, synthesis (using 3D-NSSA models) and change detection are demonstrated. Filtering and tracking are studied in detail and significantly improved performance over related work is demonstrated. We have also successfully demonstrated the use of the NSSA for model-based compression of landmark shape sequence data. In addition, the landmark shape based techniques discussed in this chapter has been extended to steganographic applications for hiding binary information inside structured shapes like text characters and glyphs [2] (more details in [83, 84]. A patent has been filed : US Patent No. 12/650,289).

[2]This work was performed at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA

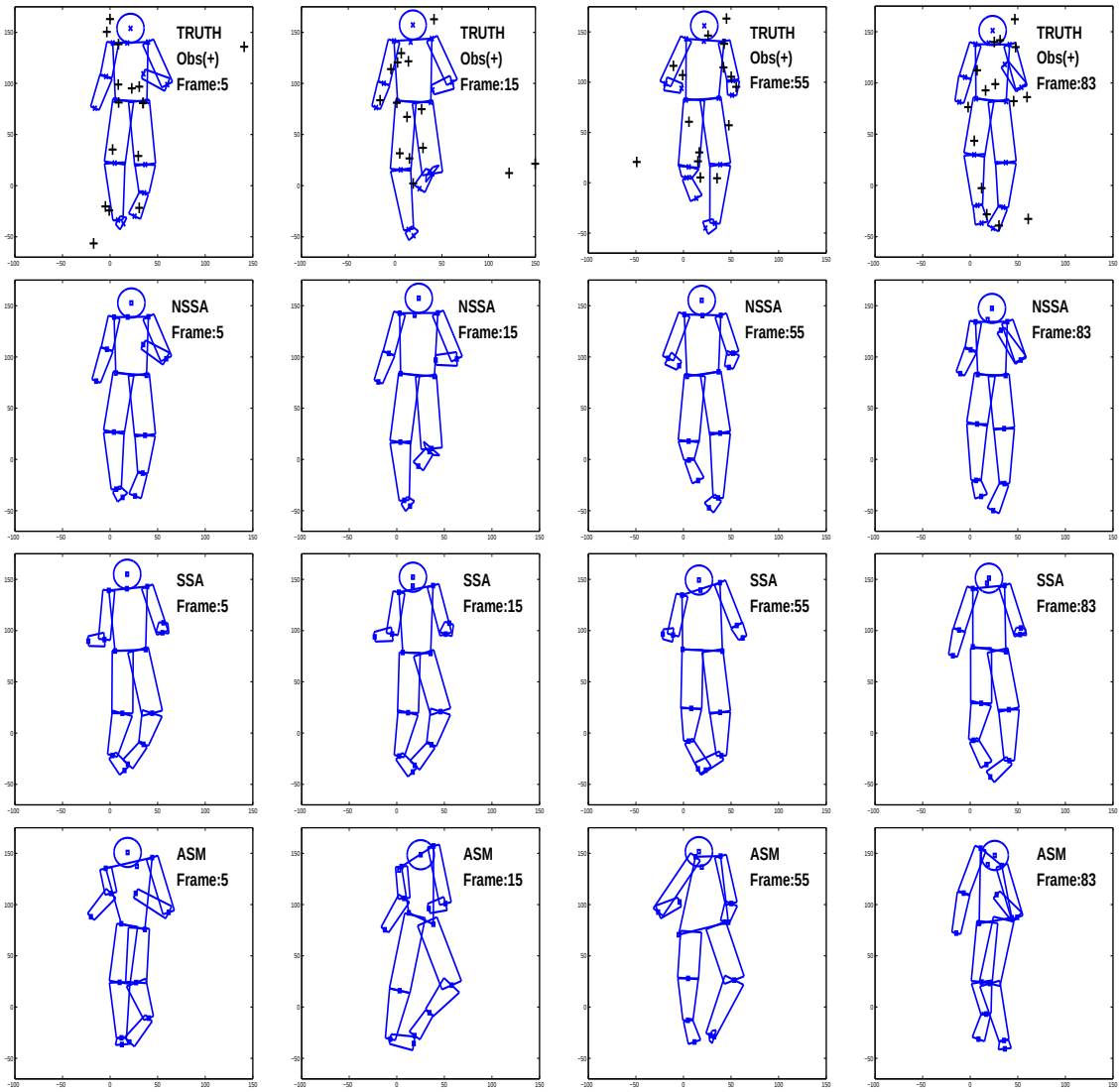


Figure 3.19 Filtering out true shape data from noisy/cluttered observations using PF-Gordon (motion activity : *run*). The landmark points (denoted by $\times$ for ground truth and $\square$ for the filter output) are fitted with rectangular patches to visualize the body posture at a given time instant. The tracking performances of NSSA, SSA and ASM are shown over 4 frames. Top row: Ground truth with observations (+), second row: tracked with NSSA, third row: tracked with SSA and fourth row: tracked with ASM. It can be clearly seen that NSSA way outperforms SSA and ASM.

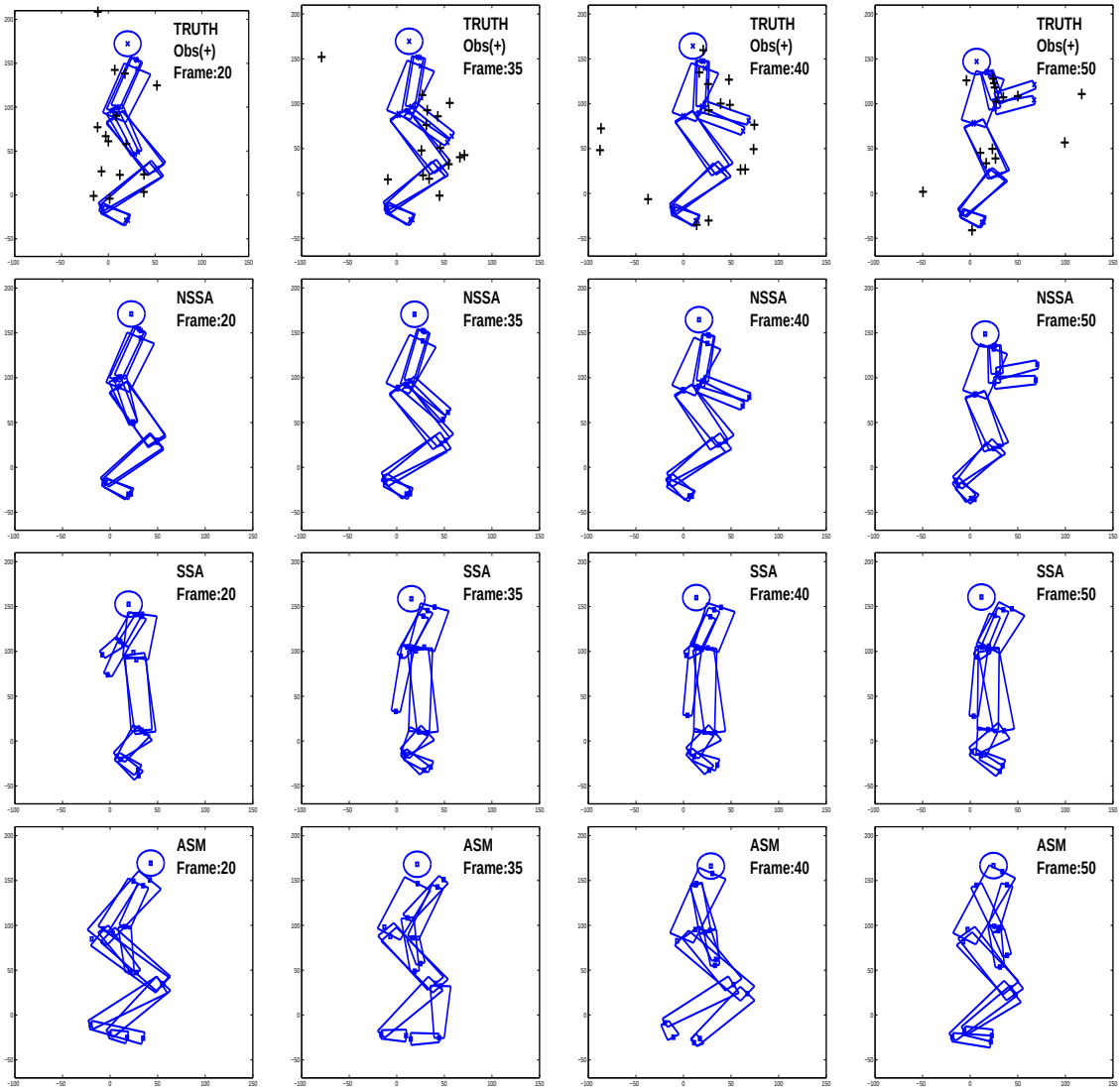


Figure 3.20 Filtering out true shape data from noisy/cluttered observations using PF-Gordon (motion activity : *jump*). The landmark points (denoted by $\times$ for ground truth and $\square$ for the filter output) are fitted with rectangular patches to visualize the body posture at a given time instant. The tracking performances of NSSA, SSA and ASM are shown over 4 frames. Top row: Ground truth with observations (+), second row: tracked with NSSA, third row: tracked with SSA and fourth row: tracked with ASM. It can be clearly seen that NSSA way outperforms SSA and ASM

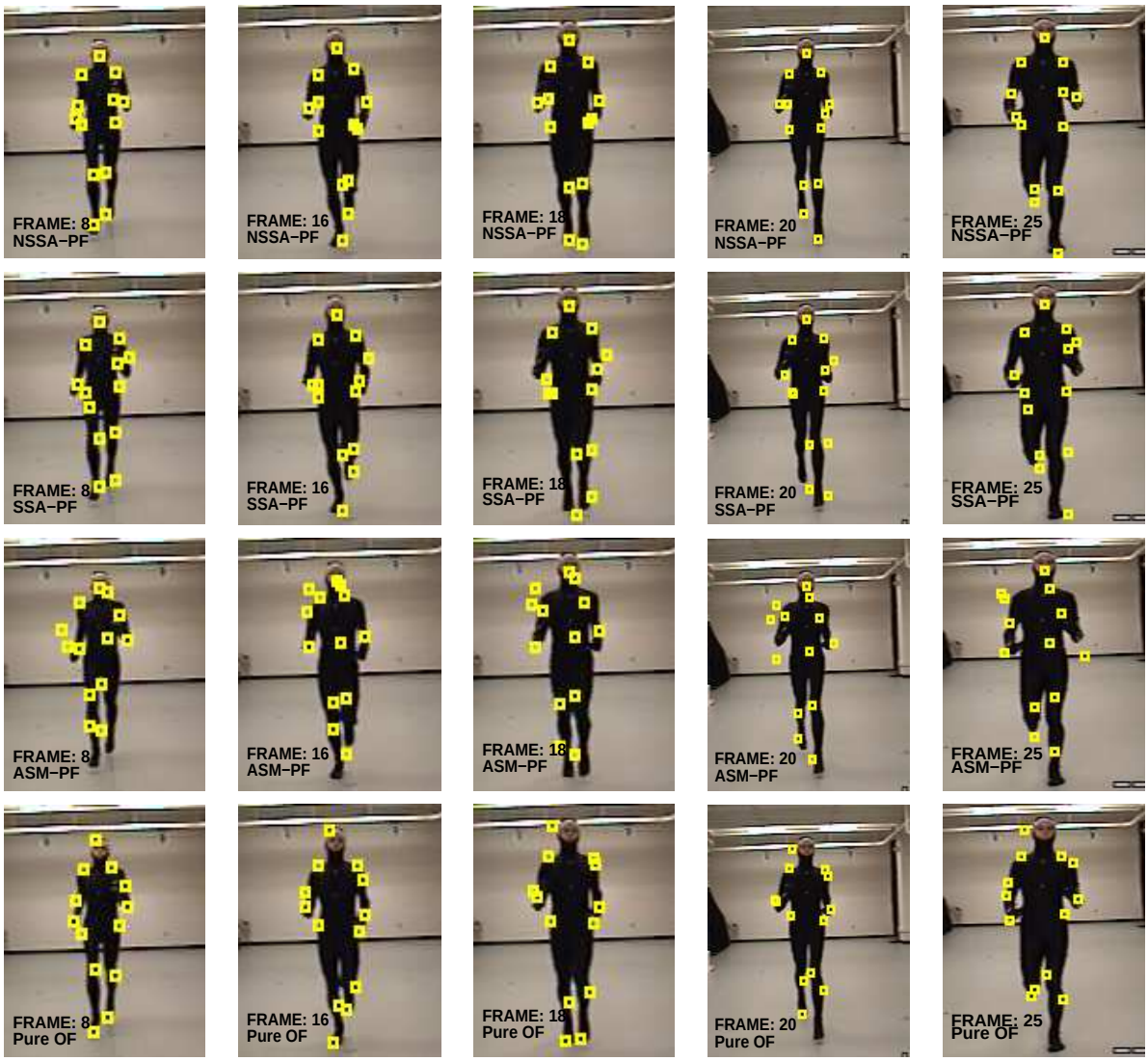


Figure 3.21 Tracking landmark configurations over the video of a running sequence (discussed in Sec. 3.4.5). Top row: NSSA, second row: SSA and third row: ASM. It can be clearly seen that NSSA way outperforms SSA and ASM. Fourth row: landmark observations extracted using purely optical flow based approach. It can be seen that the observed landmark loses posture information gradually over time. Such observation lead to poor filtering/tracking performance of the system.

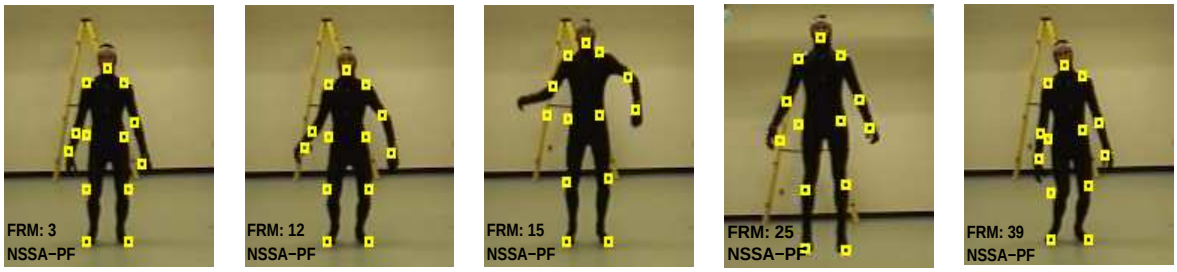
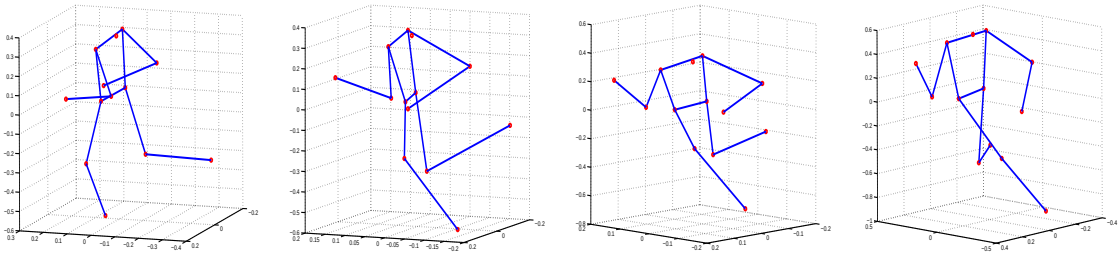


Figure 3.22 Tracking landmark configurations over the video of a jump sequence with NSSA based system model (discussed in Sec. 3.4.5). It can be seen that at frame 15, NSSA lost track of one of the landmarks (right hand). But after that it regains track of the landmark.



(a) Frame 10 (b) Frame 45 (c) Frame 51 (d) Frame 72

Figure 3.23 Synthesis of a 3D shape sequence corresponding to motion activity *run*. Four frames are shown. The system model was learnt using 3D-NSSA on 3D landmark shape sequences for running. The sequence shown above visually resembles a running sequence.

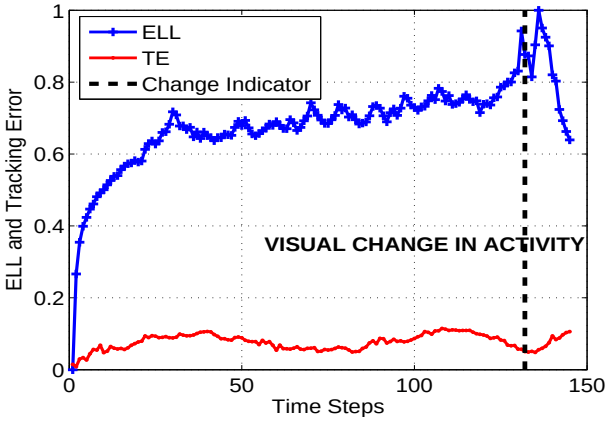


Figure 3.24 The ELL and tracking error (TE) plot for the shape sequence with *run* followed by *leap*. PF-Gordon was used with NSSA based system model. The actual activity transition occurred at $t = 132$. It can be clearly seen that ELL detected the change. There is distinct spike of ELL around $t = 132$. However, the tracker still keeps tracking even after the change has occurred.

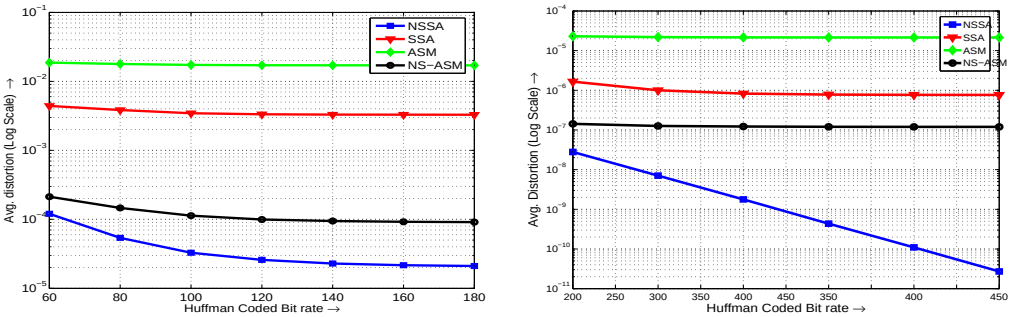


Figure 3.25 Results from 2D shape data with average distortion in log-scale. It shows that the distortion for NSSA is of the order of 10^{-5}, whereas the distortions corresponding to SSA and ASM are of the orders of 10^{-3} and 10^{-2} respectively and that of "nonstationary ASM" or NS-ASM is of the order of 10^{-4}. A total of 80 motion activity sequences, which included run, jump, dance etc taken from the CMU MOCAP database were used for this comparison.

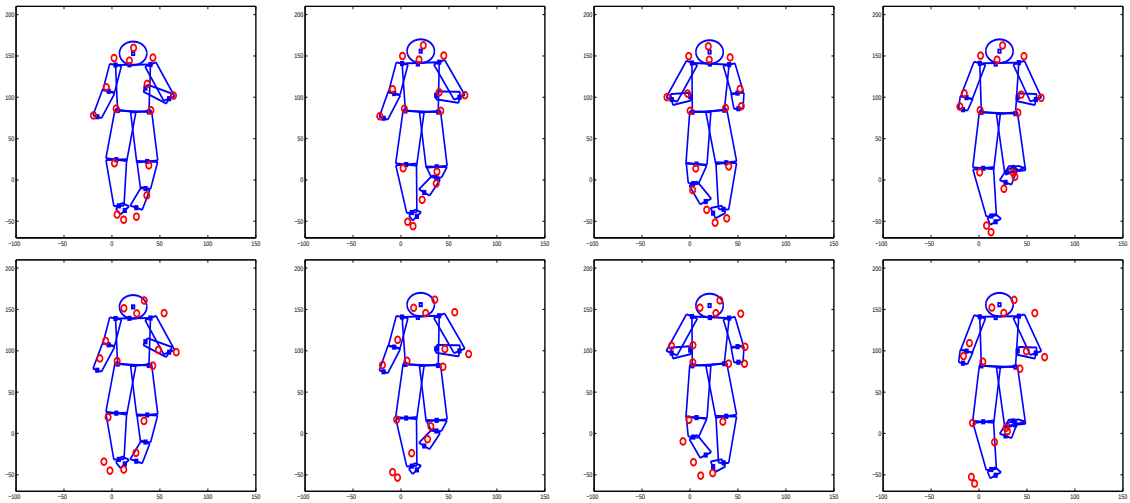


Figure 3.26 Visual comparison of reconstruction performances of NSSA and ASM. We demonstrate the original and reconstructed landmark shapes over four different frames of a video sequence corresponding to the motion activity "Run". The encoded bit rate was 120 bits/frame. The data set used in these experiments were from the CMU Mocap database. The blue squares are the original landmarks and they are fitted with rectangular patches to resemble the human body that they represent. The red circles correspond to the landmarks locations for the reconstructed shape. Top row corresponds to the NSSA based compression and the bottom row corresponds to that of ASM. As it can be seen that NSSA based compression scheme can reconstruct the body postures with reasonable accuracy. ASM, however, has worse performance compared to NSSA in this regard.

CHAPTER 4. Visual Tracking Under Illumination Changes

In this chapter, we first give a brief summary of our research contributions and related works. Then we develop a dynamical model of illumination variations in the scene and then use efficient particle filters for tracking across illumination changes under occlusion and clutter. This demonstrates the effectiveness of our tracking algorithms for solving real-world computer vision problems. Together with this, we also demonstrate the use of change detection algorithm for tracking across abrupt illumination variations.

4.0.1 Illumination invariant Tracking : Contributions and Related Works

The first contribution of this work [47] is to carefully design PF-MT for visual tracking across illumination change so that with as few as a 100 particles, we are able to successfully track on a 10 dimensional space. We demonstrate via exhaustive experiments that with 100 particles PF-MT is able to remain in track while all other PF based approaches and many other non PF based approaches fail. In this work, we use a simple motion model and a simple illumination model (taken from [14]). But as we explain below, both of these can be replaced by more sophisticated models taken from other work. Of course the sophisticated models may require specific type of imagery, e.g. [48] is a 3D-based approach and will require close range and high resolution videos, while since our model is simple, it is less restrictive and can use lower resolution videos such as those that occur in surveillance applications. A second contribution of this work is to demonstrate the use of the gELL statistic to detect model change before loss of track and to compensate for the change without ever losing track.

Early work on illumination modeling includes [49, 50, 51]. These methods focus on accurate models of appearance under varying illumination and their utility for object recognition.

However they typically required an explicit 3-D model of the object, which may not be possible to obtain from low resolution videos. For tracking, often simpler models are needed. The problem of tracking in the presence of illumination changes has been handled by different researchers. Hager and Belhumeur [52] deal with illumination change using the illumination subspace specific to an individual using her multiple images under varying illumination. In surveillance scenarios, getting such information for every person may be difficult and thus a data-independent basis such as the Legendre basis used in [14] is a better idea. Similarly trackers that use edge maps are also not a good idea since edge maps of images taken under different illuminations need not resemble each other [53]. Some other recent approaches to illumination invariant tracking involve adapting the template over time [54, 55, 56]. Direct adaption of the template requires careful selection of adaption parameters to avoid problems of drift [57], e.g. if you adapt when the tracker has latched onto clutter, it will lead to tracking failure. We show an example of this in Fig. 4.2 (third row). Another class of trackers include mean-shift [58] based tracking algorithms. It is unclear how an illumination prior can be incorporated in such cases. Failure of mean shift is shown in Fig. 4.2 (second row).

The work of [14] attempted to address the increased state space dimensionality by treating the illumination state as a discrete state that can take one of a finite set of possible values. But this model is very restrictive and for general surveillance applications, where illumination could vary a lot over the entire sequence (even though frame to frame changes are small) and one will need the finite set to be quite large. This, in turn, would require a large number of particles in PF for successful tracking. We demonstrate failure of this approach using 100 particles in Fig. 4.2 (last row).

Some other related work on visual tracking across appearance changes includes [48, 59, 60, 61, 62, 63]. These works use more sophisticated models for appearance change and for motion, e.g. [48] models the structure, 3D motion and illumination change (both 3D motion and illumination change contribute to appearance change) while [62] uses an active appearance model [64] coupled with cylinder head models which model the appearance change due to pose and illumination variations in an appearance PCA space. For tracking, some of these use a

PF, e.g. [59, 61, 63], while others use different approaches to find an exact or approximate maximum a posteriori (MAP) estimate for their models, e.g. [48] uses an alternating minimization approach to find some sort of approximate MAP estimate. The MAP estimate is easy to compute but it does not always minimize the mean squared error. The reason that [48] and others do not use a PF, which at least theoretically, in the limit of large enough particles, N, will approximate the MMSE estimate, is because their models have fairly large dimensions and a PF becomes impractically expensive (needs a very large N) for large dimensional problems. But PF-MT does not suffer from this issue and one can use the models from any of these works and use a PF-MT tracker, e.g. to use the model of [48], one can treat motion as the "effective basis" while treating all other dimensions as the "residual space".

In the next few sections, we develop a dynamical model of illumination variations in the scene and then use efficient particle filters for tracking across illumination changes under occlusion and clutter.

4.1 State Space Model

In this section, we describe the state space model for a visual tracking system under illumination variations. The system model consists of simple dynamical models for illumination and motion parameters. The observation model is developed for two different scenarios - when there is no occlusion in the scene and when there is occlusion over one or more frames. It is to be noted that, in all of our discussions, we assume a template matching based visual tracking system.

4.1.1 Notation

We use the following notation. The notation $vec(.)$ refers to the vectorization operator which operates on a $m \times n$ matrix to give vector of dimension mn by cascading the rows. $\mathbf{I}$ denotes the identity matrix. The Hadamard product (the '.*' operation in MATLAB) is denoted by $\odot$. The terms $\mathbf{1}$ and $\mathbf{0}$ refer to the column vectors with all entries as 1 and 0 respectively. $mean(.)$ gives the mean value of the entries of a vector. The function $round(\mathbf{Z})$

operates on a matrix $\mathbf{Z}$ and outputs a matrix with integer entries as $round(z_{i,j})$ which is the integer closest to $z_{i,j}$. The notation $\mathcal{N}(a;\mu,\Sigma)$ denotes the value of the Gaussian pdf with mean μ and covariance Σ computed at a whereas $x \sim \mathcal{N}(\mu,\Sigma)$ implies that the random variable x is Gaussian distributed with mean μ and covariance Σ. Similarly, the notation $\mathcal{U}(a;c_1,c_2)$ denotes the value of the uniform density defined over $[c_1,c_2]$ computed at a.

4.1.2 The Observation Model : No Occlusion

The target object template at time t is denoted as $T_t(i,j)$. The template image serves as the *crude* appearance model for the object of interest and can be expressed as $T_t(i,j) = \tilde{L}_t(i,j)T_0(i,j)$ [14], where $\tilde{L}_t$ is the unknown illumination image for frame t and T_0 is the initial template. The number of pixels contained in the template is denoted by M. Now $\tilde{L}_t$ can be different for each pixel or one can use a parametric model for it. One model that has been successfully been used in existing work [14] is to approximate $\tilde{L}_t$ as a linear combination of a set of $N_\lambda = 2k+1$ Legendre basis functions defined over the template. Thus $\tilde{L}_t$ can be written as,

$$\tilde{L}_t(i,j) = \sum_{n=0}^{N_\lambda - 1} (\Lambda_t)_n P_n(i,j) \tag{4.1}$$

where

$$P_n(i,j) = \begin{cases} 1 \;\; n = 0 \\ p_n(i) \;\; n = 1,...,k \\ p_{n-k}(j) \;\; n = k+1,...,N_\lambda - 1 \end{cases} \tag{4.2}$$

where $p_n(.)$ is the Legendre polynomial of n^{th} order and Λ_t is the vector of Legendre coefficients at time t with $(\Lambda_t)_n$ as it's n^{th} element. We call it the *illumination vector*. Then the scaled intensity value at a pixel of the template T_t is computed as,

$$T_t(i,j) = \sum_{n=0}^{N_\lambda - 1} (\Lambda_t)_n P_n(i,j) T_0(i,j) \tag{4.3}$$

where $dim(\Lambda_t) = N_\lambda = 2k + 1$. (4.3) can be rewritten as,

$$vec(T_t) = A\Lambda_t, \text{where}$$
$$A \triangleq [vec(T_0 \odot P_0), ..., vec(T_0 \odot P_{N_\lambda - 1})] \tag{4.4}$$

is a $M \times N_\lambda$ matrix.

Now, let $U_t = [s_t \ \tau_t^h \ \tau_t^v]^T$ denotes the motion parameter vector at time t w.r.t the initial frame. It corresponds to a three dimensional *motion* space encompassing scale s_t, horizontal translation τ_t^h and vertical translation τ_t^v.

The observation at time t, denoted by Y_t, is the current video frame. The observation equation is given as follows:

$$Y_t(ROI(U_t)) = vec(T_t) + \mathbf{n_t} = A\Lambda_t + \mathbf{n_t},$$
$$\text{with} \ \ ROI(U_t) \triangleq round([J_i U_t + \mathbf{i_0}, \ J_j U_t + \mathbf{j_0}])$$

$$J_i \triangleq [(\mathbf{i_0} - \tilde{i}_0 \mathbf{1}) \ \ \mathbf{1} \ \ \mathbf{0}], \ \ J_j \triangleq [(\mathbf{j_0} - \tilde{j}_0 \mathbf{1}) \ \ \mathbf{0} \ \ \mathbf{1}] \tag{4.5}$$

where $\mathbf{n_t} \sim \mathcal{N}(0, \sigma_o^2 \mathbf{I})$ and the matrix A is defined in (4.4). $ROI(U_t)$ is the set containing pixel indices for the location of the template in the current frame. The terms $\mathbf{i_0}$ and $\mathbf{j_0}$ are the M dimensional vectors containing the x and y coordinates of all the pixels in the initial template T_0, $\tilde{i}_0 = mean(\mathbf{i_0})$ and $\tilde{j}_0 = mean(\mathbf{j_0})$ denote the center of the initial template. It is to be noted that the equation above defining $ROI(U_t)$ models the uniform scaling and translation of the template pixels over time [1].

Thus the state vector is $X_t \triangleq [U_t; \Lambda_t]$. We define the observation likelihood as:

$$p(Y_t|X_t) = p(Y_t(ROI)|U_t, \Lambda_t) p(Y_t(ROI^c)|U_t, \Lambda_t)$$
$$= \frac{1}{\sqrt{2\pi}\sigma_o M} \exp(-\frac{||Y_t(ROI) - A\Lambda_t||^2}{2\sigma_o^2}) \alpha(Y_t(ROI^c))$$

where ROI^c denotes the pixels of the image outside ROI and $\alpha(.)$ is a function that does not depend on X_t, i.e. we assume that the pixels outside the target region are unaffected by the

[1] If the object is moving towards the camera in such a way that its size is increasing over time, then $Y_t(ROI(U_t))$ subsamples the (predicted) object region in the current frame to make it the same size as that of the template.

target motion or illumination. Thus as a function of X_t,

$$p(Y_t|X_t) \propto p(Y_t(ROI)|X_t) = \mathcal{N}(Y_t(ROI); A\Lambda_t, \sigma_o^2\mathbf{I}) \tag{4.6}$$

This model assumes a no occlusion scenario i.e. the region of interest (ROI) approximately contain the target object only. Under occlusion, however, this assumption will not hold and we discuss that scenario next.

4.1.3 Observation Model : With Occlusion

In case of occlusion, part or all of the ROI will be covered (occluded) by some other object(s). In the absence of any knowledge about the occluding object(s)'s intensity or pixel locations, we assume a simple outlier noise model [13]. At any time t, any ROI pixel gets occluded with probability $(1-\theta)$ independent of the others and in that case its intensity is uniformly distributed between 0 to 255 independent of all other pixels. Thus we have the following observation likelihood,

$$\begin{aligned} p(Y_t|U_t, \Lambda_t) &\propto p(Y_t(ROI)|U_t, \Lambda_t) \\ &= \Pi_{n=1}^{M}[\theta\, \mathcal{N}([Y_t(ROI)]_n; (A\Lambda_t)_n, \sigma_o^2) + \\ &(1-\theta)\mathcal{U}([Y_t(ROI)]_n; 0, 255)] \end{aligned} \tag{4.7}$$

where $[\,]_n$ denotes the n^{th} element of a vector. Note that the outlier noise term in (4.7) does not depend on X_t and thus each of the M terms in the product is a heavy-tailed pdf. The negative log-likelihood for $M = 1$ is plotted in Fig. 4.1.

4.1.4 The System Model

The state vector consists of the illumination vector and the motion parameters (scale and translation), i.e.

$$X_t = \begin{bmatrix} U_t \\ \Lambda_t \end{bmatrix} \tag{4.8}$$

where the motion parameter vector $U_t = [s_t \;\; \tau_t^x \;\; \tau_t^y]^T$ is defined in the previous section. As mentioned earlier, the illumination vector $\Lambda_t \in \mathcal{R}^{N_\lambda}$ correspond to the coefficients of the Legn-

dre basis function. Thus tracking is performed over a $N_\lambda + 3$ dimensional *motion-illumination* space.

In the absence of specific information about the object motion or about illumination variation, we assume a simple random walk model on both U_t and Λ_t i.e.

$$\begin{aligned} U_t &= U_{t-1} + n_u,\ n_u \sim \mathcal{N}(0, \Sigma_u) \\ \Lambda_t &= \Lambda_{t-1} + n_\Lambda,\ \ n_\Lambda \sim \mathcal{N}(0, \Sigma_\Lambda) \end{aligned} \tag{4.9}$$

where Σ_Λ is a $N_\lambda \times N_\lambda$ and Σ_u is a 3×3 diagonal covariance matrix.

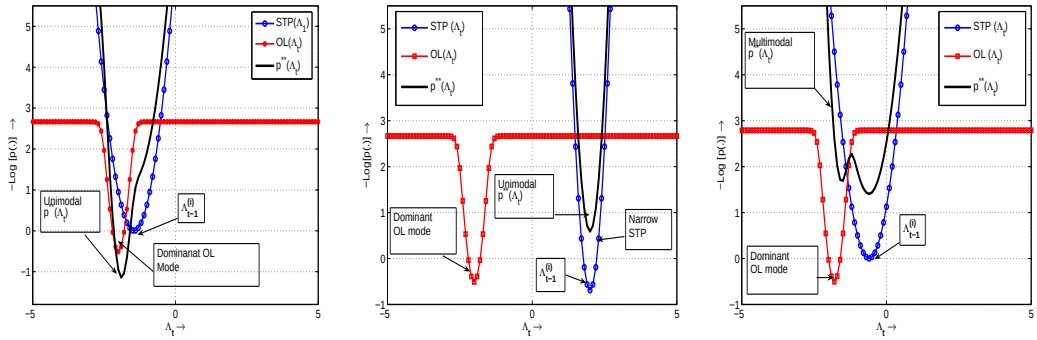
4.2 Illumination PF-MT

As discussed in Chapter 1, owing to the complexity of illumination distribution on the object, the problem of illumination tracking tends involve large dimensional state space. In addition, in order to tackle real-life computer vision challenges like occlusion and clutter the corresponding observation likelihood of the tracker also becomes multimodal. Hence, in this section, we design Particle Filter with Mode Tracker (PF-MT) for our problem. We use the efficient particle filtering framework developed in Chapter 2. We develop PF-MT for our illumination tracking problem as follows. For more details on PF-MT please refer to Chapter 2.

4.2.1 Illumination PF-MT without occlusion model

For ease of understanding, we first discuss the no-occlusion case. In the no occlusion case, the observation likelihood is unimodal conditioned on $U_t^{(i)}$ (see observation model (4.6)) and so Assumption 1 always holds (see chapter 2 for details). Also, the covariance of illumination change, learnt from training data, is small enough to justify Assumption 2 (IS-MT approximation, chapter 2). Thus we can use the PF-MT algorithm described in chapter 2 with $X_{t,s} = U_t$ and $X_{t,r} = \Lambda_t$. For our state space model,

$$\begin{aligned} p^{**,i}(\Lambda_t) &= p(\Lambda_t | U_t^{(i)}, \Lambda_{t-1}^{(i)}, Y_t) \\ &\propto \alpha \overbrace{p(Y_t | U_t^{(i)}, \Lambda_t)}^{OL} \overbrace{p(\Lambda_t | \Lambda_{t-1}^{(i)})}^{STP\ of\ \Lambda_t} \end{aligned} \tag{4.10}$$



(a) Unimodal $p^{**}(\Lambda_t)$ (b) Unimodal $p^{**}(\Lambda_t)$ (c) Multimodal $p^{**}(\Lambda_t)$

Figure 4.1 In Fig. 4.1(a) we demonstrate that when the observation likelihood (OL) mode lies in the basin of attraction of the negative log of the state transition prior (STP), the resulting p^{**} is unimodal. In Fig. 4.1(b), it can be seen that when the STP mode is far from the OL mode and the OL is heavy-tailed, that also gives rise to a unimodal p^{**}. This happens when the occlusion intensity is significantly different from the object intensity and the modes will be far apart as shown in the figure. Thus we have a narrow and unimodal $p^{**}(\Lambda_t)$. In both these cases PF-MT with $X_{t,s} = U_t$ and $X_{t,r} = [\Lambda_t]$ can be used. In Fig. 4.1(c), we show the case when the OL mode is close to the STP mode (but not in the basin of attraction of the STP) and this gives rise to a multimodal p^{**}. This happens when the intensity difference between the occluding object and the target object is not large enough. The modes will be near each other as shown in the figure. Thus we have a multimodal $p^{**}(\Lambda_t)$. This case can still possibly be handled if we include $\Lambda_{t,1}$ as part of the "effective basis", $X_{t,s}$.

where the observation likelihood (OL) is defined in (4.6), and the state transition prior (STP) of Λ_t is given in (4.9). Thus $m_t^{(i)}$ can be computed as the minimizer of $-\log(.)$ of the RHS of (4.10). This turns out to be a regularized least squares problem with a closed form solution given by,

$$\begin{aligned} m_t^{(i)} &= \Lambda_{t-1}^{(i)} + (\Sigma_\Lambda^{-1}\sigma_o^2 + A^T A)^{-1} A^T D^{(i)}, \text{where} \\ D^{(i)} &\triangleq [Y_t(ROI(U_t^{(i)})) - A\Lambda_t] \end{aligned} \tag{4.11}$$

where A and $Y_t(ROI(U_t))$ have been defined in (4.4) and (4.5) respectively. Note that the multiplier of $D^{(i)}$ can be pre-computed, making this a very fast computation.

Algorithm 10 Illumination PF-MT. Going from $\pi^N_{t-1|t-1}$ to $\pi^N_{t|t}(X_t) = \sum_{i=1}^N w_t^{(i)} \delta(X_t - X_t^{(i)})$, $X_t^{(i)} = [U_t^{(i)}\ \Lambda_t^{(i)}]$

1. *Importance Sample (IS) on effective basis :* $\forall i$, sample $U_t^{(i)} = U_{t-1}^{(i)} + n_u,\ n_u \sim \mathcal{N}(0, \Sigma_u)$ (see (4.9)). Use $U_t^{(i)}$ to compute the corresponding $ROI(U_t^{(i)})$ using (4.5)
2. *Mode Tracking (MT) in residual space :* Use the current observation to get $Y_t(ROI(U_t^{(i)}))$ and compute the mode of $p^{**,i}(\Lambda_t)$ as follows,
   (a) **No occlusion model** : Compute $m_t^{(i)}$ using (4.11)
   (b) **Occlusion model** : Compute $m_t^{(i)}$ as the mode of $p^{**,i}(\Lambda_t)$ by minimizing the $-\log(.)$ of the cost function given in (4.10) where the observation likelihood(OL) is defined by (4.7) and the state transition prior(STP) is defined by (4.9)

   Generate illumination particle as $\Lambda_t^{(i)} = m_t^{(i)}$
3. *Weighting and Resampling:* Compute the weights using (4.12) and Resample
4. *State Estimation:* Reassign the weights to $\frac{1}{N}$ and compute $\hat{X}_t$ as the average of resampled particles
5. Increment t and go back to Step 1

With the above importance sampling strategy, the weighting term will be [15]

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}},\ \text{where}$$
$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} p(Y_t | U_t^{(i)}, \Lambda_t^{(i)}) p(\Lambda_t^{(i)} | \Lambda_{t-1}^{(i)}) \tag{4.12}$$

The entire procedure is summarized Algorithm. 10 . Using this method greatly reduces the weight variance, thus reducing the number of particles required for a certain accuracy (or, equivalently, improving tracking accuracy when number of particles available is small).

Notice that in this simple no-occlusion case, Rao-Blackwellized Particle Filter (RB-PF [85, 27]) can also be used, although it will involve storing and updating covariance matrices for all illumination particles. This can become expensive (and memory-intensive) if the illumination dimension, N_λ, is large. But more importantly, RB-PF cannot be used when we model occlusions in the observation model. In most practical applications, occlusions can always occur and hence in practice we always recommend using PF-MT with occlusion which is developed next.

4.2.2 PF-MT : Illumination Tracking with Occlusion

Next consider the case with occlusion. Notice that the likelihood defined in (4.7) is the product of M heavy tailed functions. Its negative log for $M = 1$ is shown in Fig. 4.1. In this scenario the posterior will be unimodal if either the likelihood mode lies in the basin of attraction of the prior mode or if it lies far enough from the prior mode so that the distance between the two modes is "large" compared to the prior variance. If the distance is somewhere in between, it will result in a multimodal posterior. We show the three scenarios in Fig. 4.1.

In our application, the prior mode is the target's illumination at $t-1$. The first scenario above corresponds to the no occlusion case, i.e. when the dominant likelihood mode is the target's illumination at the current time plus some Gaussian noise. Since illumination changes slowly, both the prior and the likelihood modes will be close and hence this results in a unimodal $p^{**}(\Lambda_t)$ (Fig. 4.1(a)). The second scenario above corresponds to occlusion, but with the occluding object's intensity (dominant mode of the likelihood) being different enough from the target's illumination vector ("different enough" in comparison to the illumination change variance). Since the illumination change variance is small, this would happen in most occlusion cases. Thus, the third scenario (occluding object's intensity being close to the target's intensity) is much less likely than the second one.

4.3 Illumination PF-MT with Illumination Model Change

In most cases, the illumination changes very gradually over time and hence the illumination change variance takes a small value. The exception is when a car or a person transitions from shadow to sunlight or vice versa or in an indoor scenario if the light bulb is switched off or on. During these transitions, if we track with a small illumination variance model, the tracker will gradually lose track. Thus there is a need to detect model change and to assign a high illumination change variance temporarily during the transition period and to change it back once the transition is over [2]. We propose to detect model change using the recently proposed

[2]Clearly one cannot keep the illumination variance high all the time since that would make the tracker very sensitive to observation noise or occlusions (outliers) and would disallow use of the IS-MT approximation (which helps to track accurately with few particles).

generalized Expected (negative) Log Likelihood (gELL) statistic [16]. As explained in [16], gELL is designed to detect model changes before complete loss of track, which is what our application needs. In fact it works by using the partly tracked part of the change to detect it.

4.3.1 gELL statistic and its computation

Generalized ELL (gELL) is the Kerridge inaccuracy [86] between the posterior at time t, $\pi_{t|t}$ and the Δ-step ahead prediction distribution $\pi_{t|t-\Delta}$, i.e.

$$gELL(t,\Delta) \triangleq -E_{\pi_{t|t}}[-log\pi_{t|t-\Delta}(X_t)] \tag{4.13}$$

where $E_p[.]$ denotes expectation w.r.t pdf p. In practical applications it is not clear how to choose Δ. One option is to compute the maximum of gELL over all Δ, i.e. to compute

$$\text{gELL-max}(t) \triangleq \max_{\Delta=1,2,\ldots,t} gELL(t,\Delta) \tag{4.14}$$

gELL can be computed for the entire state vector or for a part of it. In this work we compute it for the illumination state, Λ_t, since we want to detect illumination model change. To compute gELL, we need a closed form approximation of $\pi_{t|t-\Delta}$. To get that, we need to approximate the PF estimate of the posterior at $t-\Delta$, $\pi^N_{t-\Delta|t-\Delta}(X_t)$, in closed form. In this work, we just use a Gaussian density approximation i.e. $\pi^N_{t-\Delta|t-\Delta}(X_t) \approx \mathcal{N}(X_t\ ;\ \mu^N_{t-\Delta|t-\Delta}, \Sigma^N_{t-\Delta|t-\Delta})$ where the parameters are estimated as the empirical mean and covariance of the weighted particle set for $\pi^N_{t-\Delta|t-\Delta}(X_t) = \sum_{i=1}^N w^{(i)}_{t-\Delta}\delta(X_t - X^{(i)}_t)$. With this approximation, the prediction density, $\pi_{t|t-\Delta}(X_t)$, which is obtained by applying the system model of Λ_t, given in (4.9), Δ times to $\pi_{t-\Delta|t-\Delta}(X_t)$, is also Gaussian i.e. $\pi_{t|t-\Delta}(X_t) \approx \mathcal{N}(X_t\ ;\ \mu^N_{t|t-\Delta}, \Sigma^N_{t|t-\Delta})$ with the mean and covariance defined in (4.15) below. Thus, gELL is computed as:

$$gELL(t,\Delta) \triangleq \sum_{i=1}^{N} w^{(i)}_t (\Lambda^{(i)}_t - \mu^N_{t|t-\Delta})^T (\Sigma^N_{t|t-\Delta})^{-1} (\Lambda^{(i)}_t - \mu^N_{t|t-\Delta})$$

$$\begin{aligned}
\mu^N_{t|t-\Delta} = \mu^N_{t-\Delta|t-\Delta} &\triangleq \sum_{i=1}^{N} w^{(i)}_{t-\Delta} \Lambda^{(i)}_{t-\Delta} \\
\Sigma^N_{t|t-\Delta} &\triangleq \Sigma^N_{t-\Delta|t-\Delta} + \Delta\Sigma_\Lambda \text{ and} \\
\Sigma^N_{t-\Delta|t-\Delta} &\triangleq \sum_{i=1}^{N} w^{(i)}_{t-\Delta} (\Lambda^{(i)}_{t-\Delta} - \mu^N_{t|t-\Delta})(\Lambda^{(i)}_{t-\Delta} - \mu^N_{t|t-\Delta})^T
\end{aligned} \tag{4.15}$$

As explained in [16], the threshold for detecting model change can be set at a value that is a little above $E_{Y_{t-\Delta+1:t}}[gELL|\text{no change}]$ which is equal to $E_{\pi_{t|t-\Delta}}[-\log \pi_{t|t-\Delta}(\Lambda_t)]$. Since $\pi_{t|t-\Delta}$ is approximated by a Gaussian, this turns out to be equal to the dimension of Λ_t plus a constant (which is the same in both the gELL computation and the threshold computation and hence is ignored).

4.3.2 Illumination PF-MT with Change Detector

We begin by running the Illumination PF-MT algorithm of Algorithm. 10 with Σ_Λ given by the learnt illumination covariance. At each time t, after the weighting step, we compute gELL as described above. If it exceeds a threshold, then we set Σ_Λ to a heuristically selected large value. During this period the tracker almost exclusively relies on the observations. Assuming no occlusion during this transition period, the particles will quickly and correctly adapt to the changed illumination conditions. At this point, the gELL statistic value will reduce. When it goes below the threshold, we reset Σ_Λ to its learnt value.

4.4 Experimental Results

We began by first comparing the tracking performance of our method (Illumination PF-MT) with other trackers (both PF and non-PF based) for simple no-occlusion scenarios. These methods used models different from ours. Then we compared Illumination PF-MT with some other PF algorithms using the same model as ours which include - a) PF-Gordon [2], b) Auxiliary PF [87], and c) PF-Gordon without an illumination model, both for videos with and without occlusion. Finally, we evaluated the ability of our algorithm to automatically detect and adapt to shadow-light transitions.

We used a set of labeled video sequences for learning the observation model and the dynamical models on Λ_t and U_t. For learning the illumination model, we used $N_\lambda = 7$ (i.e. used Legendre functions up to the order of 3). For all the PF algorithms, we used a fixed particle size of $N = 100$. Since most of the real-life trackers would have to deal with occlusions, we use the occlusion model (4.7) as our generic observation model for most of the experiments, i.e. we used Algorithm 10 with step 2b (occlusion case).

4.4.1 Face Tracking under Illumination Change without Occlusions

In this experiment, the test data contained no occlusions and we used the observation model without occlusion given in (4.6). The results are shown in Fig. 4.2. The box corresponds to the estimate (MMSE estimate in case of PF methods) of the state vector, U_t, given by the algorithm. The top row shows an individual tracked successfully using Illumination PF-MT (our method) with 100 particles. The second row shows tracking failure while using mean shift tracking [58] which is a model free approach. An alternative to using an illumination model is to adapt the template periodically as discussed in [54, 55, 56]. In our experiments, we adopted a simple idea for template adaptation, namely updating the template every T_{adapt} frames. We show the result while using $T_{adapt} = 5$ in the third row. When the background clutter is similar to the object, this leads to drift and tracking failure. Similar results were seen for $T_{adapt} = 1, 10$ as well (not shown). In the fourth row we demonstrate failure of the algorithm of [14], which treats the illumination state as a discrete state that could take one of a finite set of possible pre-learnt values, when using only 100 particles. Some other instances of face tracking using our system has been shown in Fig. 4.3 and Fig. 4.4.

4.4.2 Face Tracking under Illumination Change and Occlusions

First, we tested Illumination PF-MT for tracking across minor occlusions under variable lighting conditions. The occlusions were introduced when the person covered his face with a book for a couple of frames. The lighting conditions variations could be attributed to two factors - a) the target's distance from the window and variable ambient lighting coming through

it, and b) occasional switching off of the light sources inside the room. The tracking results are shown in Fig. 4.5.

In a second set of experiments, we compared the performance of Illumination PF-MT with other PF methods while tracking across severe occlusions and under the same variable illumination conditions as above. The occlusions in these cases were introduced when another person walks across the camera blocking the target. The visual comparison results are shown in Fig. 4.6. It can be clearly seen that the Illumination PF-MT (top row) remains in track before the occlusion and is able to recover from the partial loss of track due to the occlusion. On the other hand, all the other methods lose track even before the occlusion starts. The reasons are discussed in detail in the next subsection.

4.4.3 Vehicle Tracking under Illumination Change

The car dataset was generated from a camera observing a road from above as cars approach an intersection. The illumination variations were due to variation in the ambient lighting conditions. We compared the tracking performance of Illumination PF-MT (implemented with the occlusion model) with other methods such as PF-Gordon [2], Auxiliary-PF [87] etc. The quantitative tracking accuracy plots are given Fig. 4.7. It can be seen that Illumination PF-MT significantly outperforms the rest. PF-Gordon with illumination model performs the worst; even worse than the case when we do not have a model for illumination at all. This can be attributed to the low effective particle size of PF-Gordon [2] which tracks on a 10 dimensional space where as PF-Gordon-without-illumination-model tracks on a 3 dimensional space. A particle size of 100 was sufficient for PF-MT, because it utilized an efficient importance sampling technique that sampled over a 3-dimensional effective basis only. PF-Gordon fails because - 1) it performs importance sampling over 10 dimensions which necessitates large particle size (much larger than 100) for a reasonable tracking accuracy and 2) it does not use the current observation for importance sampling and solely relies on the system model. The use of auxiliary resampling strategy in the Auxiliary PF [87] facilitates a minor performance improvement but it is still much worse than Illumination PF-MT. The PF with no illumination

model remains in track for sometime but eventually loses track as the template appearance changes beyond a certain point.

We then compared the tracking performances of Illumination PF-MT with and without occlusion model (i.e. using the observation models (4.7) and (4.6) respectively). A simulated occlusion (simulated by generating i.i.d. uniform(0,255) noise) was introduced at the 12th frame and removed at 14th frame. Illumination PF-MT without an occlusion model fails in the presence of occlusion. Illumination PF-MT with an occlusion model, however, keeps tracking all along. We show the visual tracking comparisons in Fig. 4.8. The top row demonstrate that Illumination PF-MT with occlusion model maintains tracking across occlusions. However, Illumination PF-MT without occlusion model fails (second row).

4.4.4 Dealing with Illumination Model Change

Finally, we demonstrate the utility of the change detection based tracking framework. The car dataset was generated from a camera observing a road from above as cars approach an intersection and move in and out of shadow. In Fig. 4.9 we show the results of using gELL and Illumination PF-MT using 100 particles. Around frame 19, when the car starts to move from sunlight to shadow, the gELL value starts to increase from its sunlight value (see the gELL plot). When it goes beyond the gELL threshold (computed as explained earlier in Sec. 4.3.1), we set $\Sigma_\Lambda = \Sigma_\Lambda^{large}$. We set $\Sigma_\Lambda^{large} = \infty$ to allow the tracker to completely rely on the observations. When gELL decreases again, we reset Σ_Λ to Σ_Λ^{learnt}. The tracking results are shown in the first row of Fig. 4.9. If we do not detect the transition and hence do not increase Σ_Λ then the tracker loses track at the transition time. This is shown in the second row.

An important point to make is that if an occlusion occurs during the transition period (i.e. during model change), tracking would fail. This is because during the transition, we increase Σ_Λ to a large value so that the tracker only relies on the observations for tracking. Thus, if occlusion occurs, the template will immediately adapt to the occluding object's intensity and would not have a way to recover. Another instance of tracking cross drastic illumination changes is demonstrated on a sequence from CAVIAR dataset in Fig. 4.10.

4.5 Summary

In this Chapter, we have tackled the problem of visual tracking under variable illumination as an inference problem in a joint *motion-illumination* space introduced in [14]. We used the PF-MT [15] idea to exploit the fact that given the motion vector at time t, the posterior of the illumination will usually be unimodal and narrow enough to allow us to replace importance sampling by mode tracking (IS-MT) for illumination. This key step ensures accurate tracking with much fewer particles than those needed by any other existing PF. We show exhaustive experiments to demonstrate the superior performance of our algorithm in handling large illumination variations and severe occlusions for both face and vehicle tracking videos. We also use the recently proposed idea of generalized ELL (gELL) to detect and adapt to changes in the illumination model and this helps remain in track across lighting to shadow type of illumination model changes.

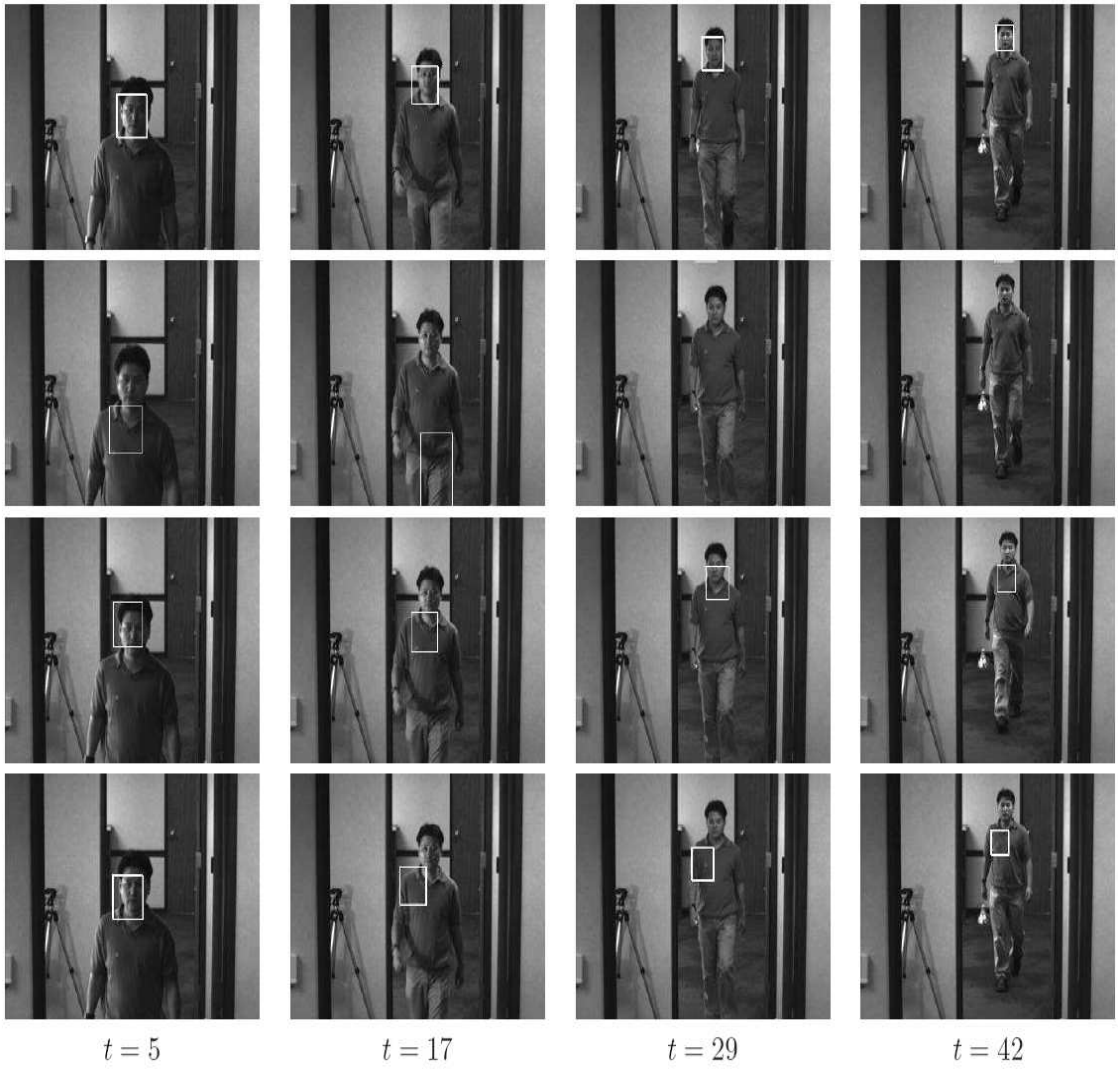
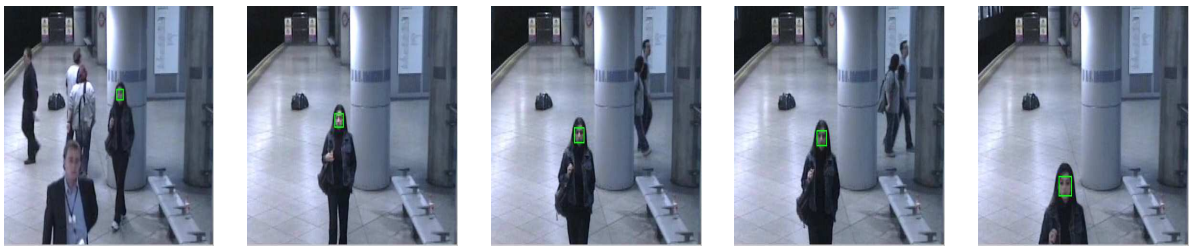
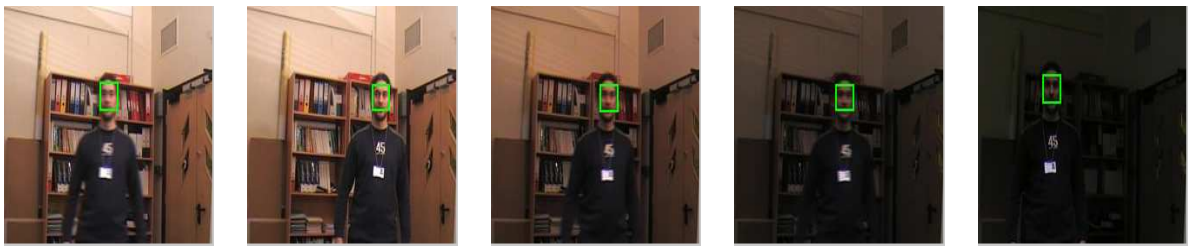


Figure 4.2 Tracking faces across illumination changes. A total of 100 particles were used in each case except second row. The top row shows a person being tracked across illumination changes using Illumination PF-MT. (b) shows the case when using mean shift tracker (c) shows the case of using an adaptive tracker with adaptation rate set to 5 frames (d) shows the case of using 6 centroids using the method of Kale and Jaynes [14].



t = 5 t = 50 t=80 t=90 t=145

Figure 4.3 An instance of face tracking for surveillance in a subway station.



t = 20 t = 90 t=100 t=110 t=130

Figure 4.4 An instance of face tracking under large illumination variation when someone switches the lighting conditions in a room.

(a) t=23 (b) t=58 (c) t=70 (d) t=78 (e) t=108

Figure 4.5 Tracking through real-life occlusion along with illumination variations using Illumination PF-MT. This sequence correspond to a minor occlusion scenario where the face was partially covered with a book for up to 8 frames.

Figure 4.6 Visual comparison of various methods for face tracking across illumination changes with occlusion lasting up to 6 frames. Again we used $N = 100$ particles. The top row corresponds to Illumination PF-MT (our method). Second row correspond to the case when no model for illumination is used. The third row correspond to Full PF with Auxiliary Resampling (called Auxiliary PF [87]). The fourth row correspond to PF-Gordon or Full PF [2] with standard resampling strategy. It can be seen that Illumination PF-MT outperforms the rest. It is to be noted that with limited number of particles ($N = 100$), PF-Gordon looses track right from the beginning. This is because, PF-Gordon fails to estimate the illumination vector correctly with insufficient number of particles.

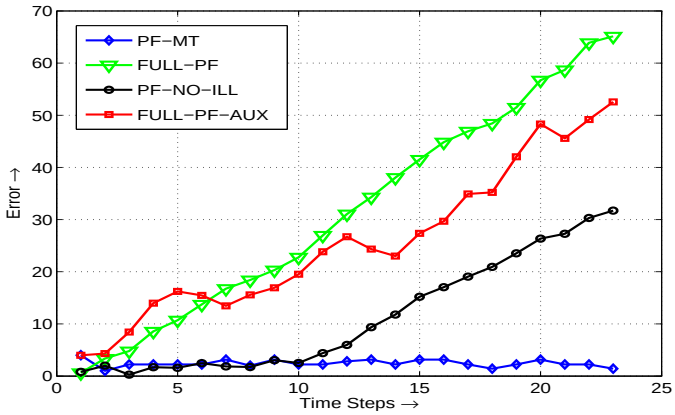


Figure 4.7 Performance comparison of various PFs while tracking across illumination changes for the car sequence. We show the location error from the ground truth for different particle filters. PF-MT correspond to Particle Filter with Mode Tracker (i.e. Illumination PF-MT), FULL-PF correspond to PF-Gordon [2], Full-PF-AUX correspond to Full PF with Auxiliary resampling strategy (Auxiliary-PF [87]). It can be seen that Illumination PF-MT outperforms the rest. It is to be noted that Auxiliary-PF has some negligible performance improvement over PF-Gordon with standard resampling strategy; but it is far worse than Illumination PF-MT. In all of these experiments we used $N = 100$ particles.

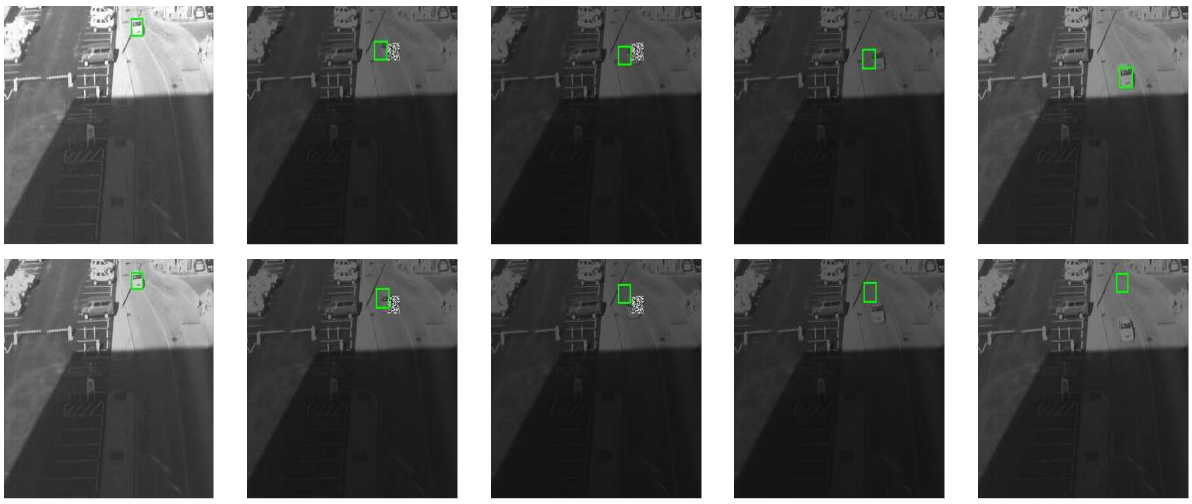


Figure 4.8 In this Figure, we compare visual tracking performances of Illumination PF-MT with and without occlusion model (i.e. using the observation models (4.7) and (4.6) respectively). The top row demonstrate that Illumination PF-MT with occlusion model maintains tracking across simulated occlusion lasting up to 3 frames (frame 12 through 14). However, the lack of an occlusion model leads to tracking failure at the event of an occlusion (second row).

t = 1 t = 19 t = 40 Using gELL

t = 1 t = 19 t = 28 t = 40

Figure 4.9 This Figure shows the results using gELL based change detection statistics. It can be see that the tracker can track through drastic illumination changes. The top row demonstrates that we are able track through illumination model changes when the car moves from sunlight to shadow area. During the transition from sunlight to shadow area (around frame 19), the gELL value goes above threshold indicating a model change (see gELL plot in the top row). If we do not detect the transition and increase Σ_Λ then, tracker fails (second row).

t = 20 t = 30 t=40 t=60 t=90

Figure 4.10 An instance of tracking cross drastic illumination changes is demonstrated on a sequence from CAVIAR dataset.

CHAPTER 5. Particle Filtered Modified Compressive Sensing

In this chapter, we propose a novel algorithm for recursive estimation of a sparse signal sequence. In our method, the idea of recently proposed regularized modified compressive sensing (reg-mod-cs) is merged with sequential Monte Carlo techniques like Particle Filtering. Reg-mod-CS facilitates sequential reconstruction of the sparse signal at each instant, provided a partial knowledge of the support and the signal estimate on that support at the previous instant. We call it Particle Filtered Modified Compressive Sensing or PaFiMoCS. The sequential Monte Carlo step allows the system to render various possibilities of the current support and choose the one which is most likely given the current observations. The algorithm is similar in spirit to Particle Filter with Mode Tracker (PF-MT), where the support can be considered to be the *effective basis* whereas the signal values on the current support can be considered to be the *residual space*; the difference being the fact that in our algorithm, the mode tracking step is replaced by the reg-mod-CS for each particle. We compare our algorithm with other techniques like traditional particle filtering [2, 3], modified compressed sensing [17], regularized-modified CS without PF, compressive sensing (non-sequential static case)[71, 72], weighted l1 [76] and PF-MT (support particles being independent of observation and MT replaced by reg-modCS). In our experiments, we demonstrate with simulated sparse signal sequences that our PaFiMoCS algorithm outperforms all the other methods when it comes to sequential reconstruction from a small number of random linear measurements.

5.1 Motivation

Our primary goal is causal and recursive reconstruction of a time sequence of sparse signals. We assume a slowly changing sparsity pattern over time. Also, we want our algorithm to use

as few linear measurements at each time as possible. The "recursive" aspect of the algorithm is important i.e. use current measurements and previous reconstruction to get current reconstruction. Potential applications of such an algorithm could be realtime dynamic MRI, single-pixel imaging etc. with faster acquisition(fewer measurements) and faster reconstruction(recursive).

5.1.1 Related Works

As far as sparse reconstruction is concerned, many practical approaches (polynomial complexity in signal dimension) have been proposed recent years which include greedy methods like Orthogonal Matching Pursuit [66, 67], CoSaMP[68] and also convex relaxation approaches, e.g. Basis Pursuit(BP)[69], Basis Pursuit De-Noising(BPDN) [69, 70] to name a few. Some related works from Compressed Sensing (CS) literature include [71, 72]. Sequential sparse signal reconstruction with partial knowledge of the support was proposed in [17, 18]. But that relies on slow support changes and do not use any dynamical model on the sparsity pattern change. Some other related work on sequential sparse reconstruction include Kalman Filtered Compressive Sensing [73] and sequential compressed sensing in sparse dynamical systems [74], [75]. Some other approaches include static approach like weighted l-1 [76] and static Bayesian approaches - [77, 78].

5.1.2 Sparse Reconstruction

Say, we want to reconstruct a sparse signal x, with support N, from $y := \Phi x$, when $M = \text{length}(y) < d = \text{length}(x)$. This problem can be solved under certain conditions [71, 72] if we can find the sparsest vector satisfying $y = \Phi x$. The solution can be found via exhaustive search; but that leads to exponential complexity. Many practical approaches (polynomial complexity in d) have been proposed in recent years to solve this problem [68, 72, 71, 78, 69, 70].

Now, consider the problem of sparse reconstruction with partly known support. Say, we want to reconstruct a sparse signal, x, with support, N, from the observation $y := \Phi x$, given partial but partly erroneous support "knowledge": T. Rewrite $N := \text{support}(x)$ as $N = T \cup \Delta \setminus \Delta_e$. Here, $\Delta := N \setminus T$: misses in T (unknown) and $\Delta_e := T \setminus N$: extras in T

(unknown). If $N = T \cup \Delta$, above problem is equivalent to finding the signal that is sparsest on T^c such that data constraint is satisfied. Modified-CS [17, 18] attempts to solve the problem as follows,

$$\min_{\beta} ||(\beta)_{T^c}||_1 \text{ s.t. } y = \Phi\beta$$

Exact reconstruction conditions for modCS are much weaker than that of CS when $|\Delta| << |N|$ and $|\Delta_e| << |N|$.

5.2 Problem Formulation and Notation

Say, $\{x_t\}_{t\geq 0}$ is a sparse signal sequence with $x_t \in \mathcal{R}^d$ and $d << K$, $\forall t = 0, 1, \ldots$ where K is the number of non-zero entries in x_t. The observations $\{y_t\}_{t\geq 0}, y_t \in \mathcal{R}^M$ are generated from the following observation model,

$$y_t = \Phi x_t + n_t \tag{5.1}$$

where Φ is a $M \times d$ random matrix with entries as zero-mean i.i.d Gaussian and the observation noise is normal distributed as $n_t \sim \mathcal{N}(\mathbf{0}, \Sigma_o)$. Under this set up, our primary objective is to sequentially reconstruct the signal x_t from the observations. In order to do that we develop a recursive algorithm which causally reconstructs the signal x_t from the observation y_t and signal estimates from the previous instant. Also, it is assumed that the sparsity pattern of x_t follows a certain dynamical model as described below. But first, we give some of the basic notations that we use in the sections to follow.

Denote the sparse signal at time t as x_t. Define the support of the signal x_t as, $N_t = \{i : (x_t)_i \neq 0\}$, $(x_t)_i \triangleq i^{th}$ component of x_t. $(x_t)_{N_t}$ denotes a vector comprising of the components of x_t corresponding to the indices belonging to the set N_t. The term N_t^c is given as, $N_t^c = U \setminus N_t$ where $U = [1, ..., d]$ with 'd' being the dimension of x_t. The symbol '$\setminus$' denotes *set minus*. While going from $t-1$ to t, the set of new elements being added to the support is denoted as A_t, whereas the set of deleted elements is denoted as R_t. Thus $A_t = N_t \setminus N_{t-1}$ and $R_t = N_{t-1} \setminus N_t$. The function $dim(.)$ gives the dimension of a vector. We define the state of

the system at t as : $X_t = [x_t, N_t]$. Next, we describe the generative model for the sparse signal sequence.

5.2.1 Signal Dynamics : The Generative Model

The generative model for the sparse signal sequence is given as follows.

1 At $t = 0$, start with some N_0 with $dim(N_0) << d$. Get x_0 as follows,

$$(x_0)_{N_0} = \boldsymbol{\nu_0}, \;\; \boldsymbol{\nu_0} \sim \mathcal{N}(\mathbf{m_0}, \Sigma_0)$$
$$(x_0)_{N_0^c} = \mathbf{0}$$

Thus for $t = 0$, obtain $X_0 = [x_0, N_0]$.

2 For $t > 0$ compute,

$$N_t = (N_{t-1} \cup A_t) \setminus R_t$$

where the sets A_t and R_t are generated as follows. Now, various elements in A_t and R_t are assigned as,

$$A_t \sim \text{Unif}_p(N_{t-1}^c)$$
$$R_t \sim \text{Unif}_p(N_{t-1}^*)$$
$$\text{with} \;\; N_{t-1}^* = \{j : |(x_{t-1})_j| < \Delta, j \in N_{t-1}\} \tag{5.2}$$

where $p << |N_{t-1}|$ and Δ is a heuristically chosen threshold. Finally, generate the signal x_t as,

$$(x_t)_{N_t} = (x_{t-1})_{N_t} + \boldsymbol{\nu}, \text{ where, } \boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \Sigma_\nu)$$
$$(x_t)_{N_t^c} = \mathbf{0}$$

Thus, for $t > 0$ get $X_t = [x_t, N_t]$

3 Set $t \leftarrow t + 1$ and go to step 2.

This dynamical model on x_t and in turn X_t, serves as the system model in a tracking framework. Further details on this dynamical model will be explored when we describe our algorithm. The signal dynamics is demonstrated in Fig. 5.1

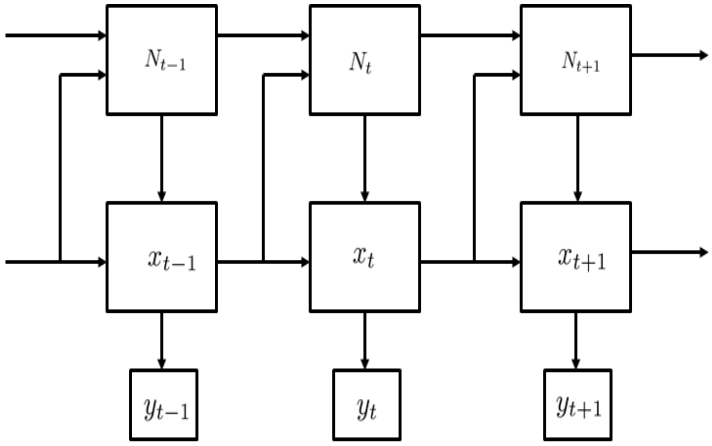


Figure 5.1 The dynamical model corresponding to the sparse signal sequence.

5.3 Signal Reconstruction : Tracking

In this section, we discuss several algorithms for estimating x_t at each instant form the observations y_t. The key question here is : given a Markov model on slow support change, and on slow nonzero signal value change, what is the "best" way to use it? One idea is Particle Filtering(PF) framework i.e. sequential importance sampling from the dynamic prior model for MAP estimates of the support and the signal.

5.3.1 Sequential Monte Carlo : PF

Basically, we are in search of an optimal estimator of x_t under the above problem formulation. One idea could be to get the Maximum Aposterior Probability(MAP) estimate of the signal (i.e. the one which maximizes $p(X_t|y_{1:t})$). Particle Filtering (PF) gives an approximate Bayesian inference of X_t under the setup. It is to be noted that form the dynamical model described above, it can be shown that $p(X_t|X_{t-1}) = p(x_t|x_{t-1}, N_t)p(N_t|N_{t-1}, x_{t-1})$ which means that the PF can render various possibilities of the support and the signal values (i.e *Importance Sampling or IS*) and assign importance weights based on the OL i.e. $p(Y_t|X_t)$. Then the MAP estimate becomes the importance sample with highest importance weight. The basic PF algorithm under this setup is given in Algorithm. 11. A mode-tracking (MT) version of the PF with a modified CS step is given in Algorithm. 12. This does not use the observations

Algorithm 11 PF : Recursive Estimation of $X_t = [N_t, x_t]$

At $t = 0$, $N_0^{(i)} = N_0$ and $x_t^{(i)} = \hat{x}_0$, for $i = 1, 2, ..., N_{PF}$

At each time $t > 0$, for $i = 1, 2, ..., N_{PF}$

(a) Imp. Sample on support : $A_t^{(i)} \sim \text{Unif}_p(N_{t-1}^{(i)\,c})$ and $R_t^{(i)} \sim \text{Unif}_p(N_{t-1}^{(i)\,*})$. Get $N_t^{(i)} = (N_{t-1}^{(i)} \cup A_t^{(i)}) \setminus R_t^{(i)}$ and define $N_t^{(i)} \triangleq T$

(b) Imp. Sample on signal value : $x_t^{(i)} : (x_t^{(i)})_{T^c} = \mathbf{0}$, $x_T^{(i)} \sim \mathcal{N}((x_{t-1}^{(i)})_T, \Sigma_\nu)$

(c) Assign importance weights :
$w_t^{(i)} \propto p(y_t|X_t) \propto \exp(-\frac{1}{2}(y_t - \Phi x_t^{(i)})^T \Sigma_o^{-1}(y_t - \Phi x_t^{(i)}))$

(d) MAP estimate : $\hat{x}_t = x_t^{(i)}$ such that $i = \arg\max_i w_t^{(i)}$

(e) Resample and move to $t = t + 1$

for generating importance samples of the support. As we shall see that the support error for these type of methods accumulate over time, making them unstable and impractical.

5.3.2 CS based reconstruction : Mod-CS,Reg-mod-CS, Weighted l1

Modified Compressive Sensing or mod-CS [17] takes a sequential approach to compressed sensing for recursive estimation of sparse signals. It has been shown in [17] that with partial knowledge of the current support N_t, the sparse signal x_t can be reconstructed with fewer number(compared to the traditional CS) of measurements by solving the following L1-minimization problem. For the noisy observation case, it becomes as modified BPDN [17].

$$\hat{x}_t = \arg\min_x ||x_{T^c}||_1 + ||y_t - Ax||_2^2 \tag{5.3}$$

where T is the prior knowledge of the support of x_t. Generally, we can use previous estimate of the support as the predicted current support i.e. $T = \hat{N}_{t-1}$ where $\hat{N}_{t-1} = \{i : |(\hat{x}_{t-1})_i| > \Delta\}$. The above algorithm works when there is *slow* change of support i.e. $dim(A_t) + dim(R_t) << dim(N_t)$ or in other words, T is a reasonably close estimate of N_t.

Under large observation noise, further constraints can be put in the above optimization problem to do a better reconstruction. Regularized-modified-CS does exactly that. It assumes

Algorithm 12 PF-MT : Recursive Estimation of $X_t = [N_t, x_t]$

At $t = 0$, $N_0^{(i)} = N_0$ and $x_t^{(i)} = \hat{x}_0$, for $i = 1, 2, ..., N_{PF}$

At each time $t > 0$, for $i = 1, 2, ..., N_{PF}$

(a) Imp. Sample on support : $A_t^{(i)} \sim \text{Unif}_p(N_{t-1}^{(i)\,c})$ and $R_t^{(i)} \sim \text{Unif}_p(N_{t-1}^{(i)\,*})$. Get $N_t^{(i)} = (N_{t-1}^{(i)} \cup A_t^{(i)}) \setminus R_t^{(i)}$ and define $N_t^{(i)} \triangleq T$

(b) Mode Track on non-zero signal values :
$x_t^{(i)} = \arg\min_x \lambda||(x - x_{t-1}^{(i)})_T||_2^2 + \gamma||x_{T^c}||_1 + ||y_t - \Phi x||_2^2$

(c) Assign appropriate importance weights and resample

(d) MAP estimate : output maximum weight particle

that the signal values in the known part of the support changes slowly over time (which is a reasonable assumption for many practical applications) and the corresponding optimization problem is modified as,

$$\hat{x}_t = \arg\min_x \lambda||x_{T^c}||_1 + \gamma||(x_t)_T - (\hat{x}_{t-1})_T||_2^2 + ||y_t - Ax||_2^2 \tag{5.4}$$

where the parameters λ and γ are chosen appropriately (for details see [17, 18]). Another approach called the weighted l-1 proposed by [76] solves the following problem,

$$\hat{x}_t = \arg\min_x \gamma||x_{T^c}||_1 + \gamma'||x_T|| + ||y_t - \Phi x||_2^2$$

It is important to note that for both mod-CS and regularized mod-CS, the reconstruction performance largely relies on how good the current support estimate is. As mentioned earlier, if it is close enough to the true support, the algorithms works well as expected. But, in case there happens to be a large support change associated with successive time instants, then $\hat{N}_{t-1}$ is no longer close to N_t and hence, the reconstruction performance degrades. Now, in order to ensure a good reconstruction accuracy, our ultimate goal would be to predict the current support which is as close to N_t as possible. This brings us to our sequential Monte Carlo based approach to this problem, which we call Particle Filtered mod-CS (PaFiMoCS). We describe it in the section to follow.

5.3.3 Particle Filtered Modified CS(PaFiMoCS)

The primary goal of PaFiMoCS is to facilitate the reg-modCS algorithm with a reasonably good estimate of the true support. In all of the previous works on sequential compressive sensing, researchers used the previous support estimate as the current predicted support. But, as stated above, this heuristic can take a severe beating when there happens to be a large support change over successive time instants. PaFiMoCS, precisely, attacks this problem. It leverages from the fact that we can do a better job in support prediction by assuming a dynamical model on the support changes. Now, even if there happens to be a large support change, we still have a better handle of the situation than just heuristically using the previous support estimate for the prediction.

A dynamical model on sparsity pattern change is a reasonable assumption as long as it is *generic* in nature. The dynamical model on support change used by PaFiMoCS is described in Sec. 5.2.1. Let us have a closer look at the dynamics of support change or sparsity pattern change of x_t. Basically, the model captures the statistical mechanism of going from N_{t-1} to N_t. Thus, it can also serve as a predictive model for N_t given N_{t-1}. As stated in the previous section, $N_t = (N_{t-1} \cup A_t) \setminus R_t$ where A_t and R_t are the set of indices corresponding to addition and deletion respectively. Now, the *generic* nature of the dynamical model would depend on how the addition and deletion sets are chosen over time. Without the loss of generality, our model used (5.2),

$$A_t \sim \text{Unif}_p(N_{t-1}^c), \quad R_t \sim \text{Unif}_p(N_{t-1}^*)$$
$$\text{with} \quad N_{t-1}^* = \{j : |(x_{t-1})_j| < \Delta, j \in N_{t-1}\}$$

which is quite true in reality. The first equation above comes from the fact that new additions to the support can come anywhere within N_t^c. The second equation revolves around the fact that whenever a component of the support is going to be deleted, it is very likely that it would have a small signal amplitude corresponding to that component. In other words, no component of the support, with 'significant' signal amplitude, gets abruptly deleted from the support. If any component were to go, it fades away slowly or gradually rather than abruptly. This

assumption is quite practical for signals evolved from natural processes for which *continuity* is an inherent characteristics.

PaFiMoCS is developed based on the dynamical model for support change as well as dynamical model on signal value change on the known part of the support (Sec. 5.2.1). As a first step towards developing our algorithm, we observe that the dimension of N_t^c could be quite large compared to $dim(A_t)$. Now, there could be ${}^{dim(N_t^c)}C_{dim(A_t)}$ possible choices for A_t. Similarly, there are several possible choices for R_t as well. Jointly, they need to represent the true addition and deletion sets with maximum possible accuracy. This led us to the use of sequential Monte Carlo techniques for solving this problem. These methods have been predominantly used in the Particle Filtering literature. Here, the hope is that for a large enough number of samples, at least one of the candidate samples would be very close to the actual addition and deletion set. The basic idea of our approach is as follows.

At each time instant, we generate several realizations (i.e. particles) of A_t and R_t. This is done using the corresponding support particle from the previous instant (i.e. N_{t-1}). Then the current support particle corresponding to N_t is generated(5.2). This predicted support, along with previous signal particle is used to perform a reg-mod-CS to generate an estimate for the current signal (this is done for each particle). The support particle is then re-generated form the signal by magnitude thresholding. This important step corrects for any residual support error in the original support particle. Hence, we end up with several realizations of the current support and corresponding signal estimates, each of which, is a candidate of being the best 'guess' about the true support and the signal at the current time instant. How do we choose the best one ? - we use importance weighting step used in particle filters for assigning favorable particles. Each particle i.e. $X_t^{(i)} = [N_t^{(i)}\ x_t^{(i)}]$ is assigned an importance weight similar to a particle filter and the one with maximum weight is used as the current state estimate. This basically boils down to a MAP estimate of the support and the signal given the observations up to the current instant. In fact, PaFiMoCS can be treated as a variant of the Particle Filter with Mode-Tracking (PF-MT [15]) with support N_t as the effective basis and x_t as the residual part. After the importance weighting step, we resample the particles, move on to the

next instant and repeat the whole process for the next time instant. The entire procedure is summarized in Algorithm. 13 and the various steps associated with it is demonstrated in Fig. 5.2.

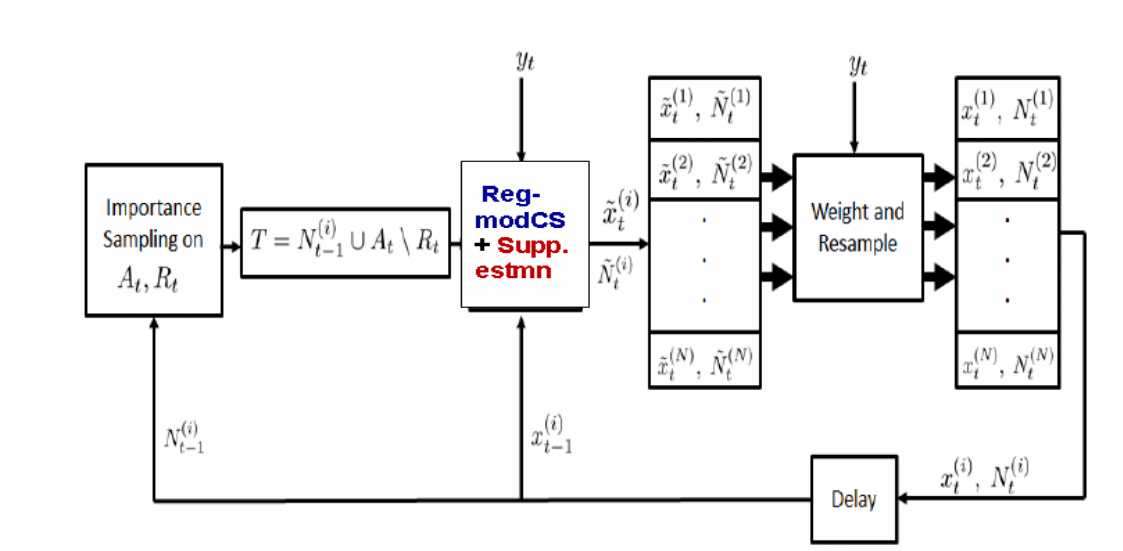


Figure 5.2 The PaFiMoCS algorithm : block diagram for various steps involved.

It is important to note that although we use a similar importance weighting step as PF-MT, there is a slight difference between our algorithm and PF-MT. In our algorithm, unlike PF-MT, we actually recompute the effective basis (i.e $N_t^{(i)}$) from the importance sample of the residual part i.e. $x_t^{(i)}$. This step is essential in our algorithm because reg-modCS step corrects if there happens to be a small error between true N_t and $N_t^{(i)}$. Thus the re-estimated support tends to be more accurate than the one generated by the importance sampling step. Otherwise, this error might accumulate over time and lead to an unstable algorithm with exploding reconstruction error (see performance plot of PF-MT).

5.4 Experimental Results and Discussion

We tested our algorithm on a simulated sequence of sparse signals. The dynamical model described in Sec. 5.2.1 was used for generating a sequence of length $T = 20$. The signal dimension was set at $dim(x_t) = d = 200$ with sparsity $K = dim(N_t) = 20$ (i.e. 10% of the signal dimension). The initial signal ($t = 0$) was generated using $N_0 = \{10, 15, 20, \ldots, 105\}$ and

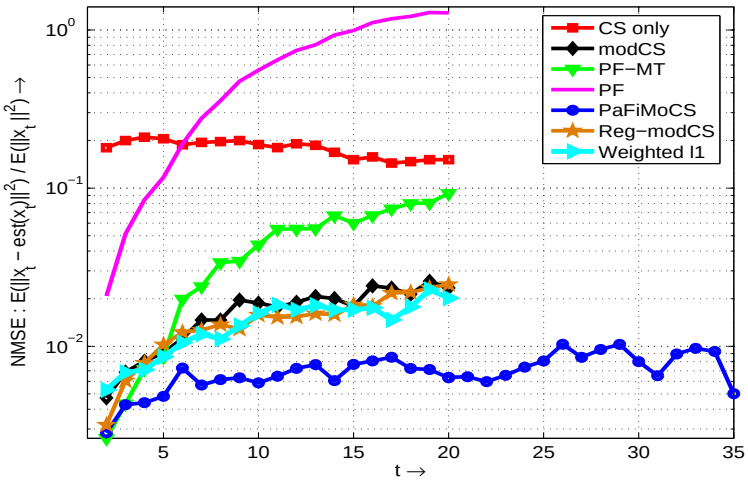


Figure 5.3 Performance comparison plots. We compute the Normalized Mean Squared Error for all the methods and compared them over 50 Monte Carlo runs. It can be seen that PaFiMoCS has the best performance among all the methods.

$\mathbf{m_0} = 5 * \mathbf{1}_{20}$ and $\Sigma_0 = 3 * \mathbf{I}_{20}$ where $\mathbf{1}_k$ is a k-dimensional column vector with all entries as 1 and $\mathbf{I}_k$ is a $k \times k$ identity matrix. We considered two additions and two deletions over successive time instants i.e. $p = 2$ or $dim(A_t) = 2$ and $dim(R_t) = 2$. The parameter Σ_ν corresponding to the dynamics of signal values was set to be $\mathbf{I}_{20}$. The entries on the observation matrix Φ was generated from a standard normal distribution with $M = 50$ (25% of the signal dimension). The observation noise was generated from a multivariate standard normal distribution i.e. $\mathcal{N}(\mathbf{0}, \mathbf{I})$ or $\sigma_o^2 = 1$ for each dimension. Under all the above parameter settings, we simulated a 200 dimensional, 20-sparse signal sequence $\{x_t\}$ and generated corresponding observation vectors $\{y_t\}$. These observations are fed to the various reconstruction algorithms to study their reconstruction performances.

We compared the reconstruction performances of seven different methods including the basic particle filter, traditional compressed sensing, modified/regularized modified compressed sensing without sequential Monte Carlo sampling, weighted l-1, PF-MT and finally PaFiMoCS. For simplicity, it was assumed that we have a perfect knowledge of the initial state of the system i.e. $\hat{X}_0 = X_0$. For regularized modCS the parameters λ and γ were set at 0.1 and 0.5 respectively. For basic particle filter and PaFiMoCS the number of particle used was

$N_{PF} = 100$. The performance of the algorithms were compared by computing the normalized mean-squared reconstruction error (NMSE) i.e. $\frac{E(||x_t - \hat{x}_t||^2)}{E(||x_t||^2)}$ for $t = 0, ..., 20$. The expectations were computed over 50 Monte Carlo runs of each algorithm. The corresponding NMSE plots for various algorithms are shown in Fig. 5.3. The comparison plots for normalized support estimation error is shown in Fig. 5.4. It can be seen that PaFiMoCS has the best performance among all the algorithms. This can be attributed to the particle filter based importance sampling step which allowed the algorithm to utilize a very accurate estimate of the supports under a reg-modCS setup, which, further corrected the residual error in the support estimate. The re-computation of the support for each particle made sure that each support particle reflects the support error correction after the reg-modCS step. That is why this algorithm has a low and stable NMSE of the order of 10^{-2}. The other variant, PF-MT which does not recompute the support for each particle (more similar to a PF-MT approach) gradually loses track of the signal as the support error accumulates over time. Other algorithms like mod-CS, reg-mod-CS and weighted l-1 without the importance sampling step, performs worse as they do not have a support prediction as good as PaFiMoCS (remember that all of them used the previous support estimate as the current predicted support). Traditional compressed sensing fails with 25% observations as it does not utilize the dynamics of sparsity pattern change as well as the constraints on the signal value changes. Basic particle filter primarily fails due to the lack of a mechanism to utilize the sparsity of the underlying signal.

Thus it turns out that by facilitating a regularized compressed sensing approach with a good support estimate from a PF framework, we can actually come up with a very efficient algorithm for sequential reconstruction of sparse signals from highly undersampled random linear measurements.

5.5 Summary

In this final Chapter, we have proposed particle filtered modified compressed sensing(PaFiMoCS) for sequential estimation (i.e. tracking) of spars signals from highly undersampled measurements. This algorithm essentially merges the ideas of compressive sensing and particle filter-

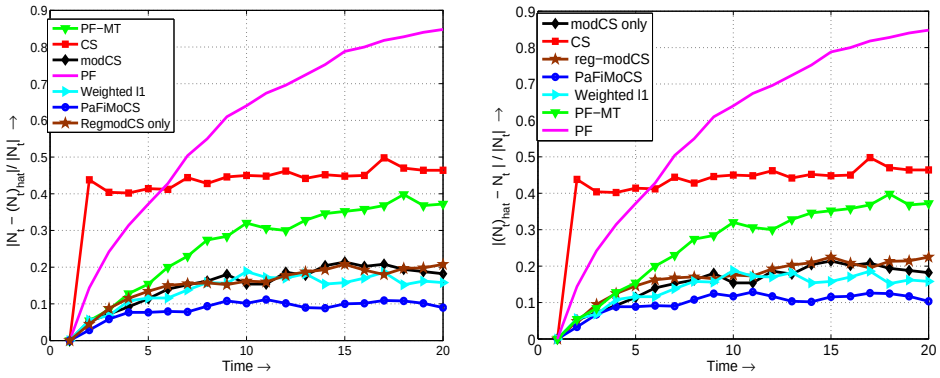


Figure 5.4 Normalized support estimation error comparison for various methods. It can be seen that PaFiMoCS has the best performance among all.

ing. The basic idea is to utilize recently proposed reg-modCS idea by providing it with a 'close enough' prediction of the support. This prediction step is carried out by sequential importance sampling technique used is particle filtering literature. The key difference between PaFiMoCS and other sequential CS based approaches is that it uses dynamical models on the support change as well as signal value change on the known part of the support. It is demonstrated with simulation experiments that PaFiMoCS performs better than other algorithms (both CS and non-CS type) which do not utilize the importance sampling step for support prediction. As a part of future research, this technique can be used in many practical scenarios where PF-MT [15] is suitable but the residual space has a slowly varying sparsity pattern over time. For example - gradual illumination variations in a scene or slowly moving occlusion in a template based tracking framework. More details to follow in the next chapter.

5.6 Appendix

For the ideal case with optimal importance sampling density [3] we have,

$$w_t \propto \frac{p(y_t|X_t)p(X_t|X_{t-1})}{p(X_t|X_{t-1}, y_t)} \tag{5.6}$$

Now, instead of the optimal density, we sample X_t from a density $\pi(.)$, which approximates the optimal importance density $p(X_t|X_{t-1}, y_t)$ as,

$$\begin{aligned} p(X_t|X_{t-1}, y_t) &= p(x_t, N_t|x_{t-1}, N_{t-1}, y_t) \\ &= p(N_t|x_{t-1}, N_{t-1}, y_t)p(x_t|N_t, N_{t-1}, x_{t-1}, y_t) \\ &\approx p(N_t|N_{t-1}, x_{t-1})\pi^*(x_t|N_t, x_{t-1}, y_t) \qquad (5.7) \\ & \qquad (5.8) \end{aligned}$$

Or, $\pi(X_t|X_{t-1}, y_t) \triangleq p(N_t|N_{t-1}, x_{t-1})\pi^*(x_t|N_t, x_{t-1}, y_t)$. Also, the state transition prior can be written as,

$$\begin{aligned} p(X_t|X_{t-1}) &= p(x_t|x_{t-1}, N_t, N_{t-1})p(N_t|N_{t-1}, x_{t-1}) \\ &= p(x_t|x_{t-1}, N_t)p(N_t|N_{t-1}, x_{t-1}) \qquad (5.9) \end{aligned}$$

Substituting (5.7) and (5.9) in (5.6),

$$w_t \propto \frac{p(y_t|x_t)p(x_t|x_{t-1}, N_t)}{\pi^*(x_t|N_t, x_{t-1}, y_t)}$$

Now, our heuristic assumption is that x_t has been sampled from a unimodal distribution $\pi^*(.)$ (use mode as the importance sample, PF-MT style). Or, $x_t^{(i)}$ is the mode of $-\log(\pi^*(.))$ (*might not be accurate*) i.e. $-\log(\pi^*(.)) \approx \lambda||x_t - x_{t-1}||_2^2 + \gamma||(x_{N_t^c}||_1 + ||y_t - Ax_t||_2^2) + \text{Const.}$ Hence, a reasonable estimate for $\pi^*(x_t|N_t, x_{t-1}, y_t)$, correct up to a proportionality constant can be given as,

$$\pi^*(x_t|N_t, x_{t-1}, y_t) \propto \exp[-(\lambda||(x_t - x_{t-1})_{N_t}||_2^2 + \gamma||(x_t)_{N_t^c}||_1 + ||y_t - \Phi x_t||_2^2)]$$

Algorithm 13 PaFiMoCS: Tracking Algorithm

Goal: From the observations $\{y_t\}_{t=0}^T$, sequentially estimate the sparse signal $\{\hat{x}_t\}_{t=0}^T$ where $y_t = Ax_t$.

1 Assume at $t = 0$, we know $\hat{x}_0$, N_0. For $i = 1 \dots N_{PF}$, assign particle set as,

$$N_0^{(i)} = N_0$$
$$x_0^{(i)} = \hat{x}_0$$

2 At $t > 0$, for $i = 1 \dots N_{PF}$:

$$(A_t^{(i)})_j \sim \text{Unif}_p(N_{t-1}^{(i)\,c})$$
$$(R_t^{(i)})_j \sim \text{Unif}_p(N_{t-1}^{(i)\,*}), \;\; N_{t-1}^{(i)\,*} = \{k : |(x_{t-1}^{(i)})_k| < \Delta, k \in N_{t-1}^{(i)}\},$$

Importance sampling on the support : $N_t^{(i)} = (N_{t-1}^{(i)} \cup A_t^{(i)}) \backslash R_t^{(i)}$. Then, using current observation y_t, perform Regularized modified CS (sort of substitute of PF-MT) to importance sample on the signal x_t as,

$$x_t^{(i)} = \arg\min_x \lambda ||(x - x_{t-1}^{(i)})_T||_2^2 + \gamma ||x_{T^c}||_1 + ||y_t - Ax||_2^2$$

where $T = N_t^{(i)}$. Recompute the support as, $N_t^{(i)} = \{j : |(x_t^{(i)})_j| > \alpha\}$. Thus get, $X_t^{(i)} = [x_t^{(i)}, N_t^{(i)}]$.

3 Assign importance weights as,

$$\begin{aligned} w_t^{(i)} &\propto \frac{p(y_t|X_t^{(i)})p(X_t^{(i)}|X_{t-1}^{(i)})}{\pi(X_t^{(i)}|X_{t-1}^{(i)}, y_t)} \\ &\propto \frac{e^{-||y_t - \Phi x_t^{(i)}||_2^2} e^{-\frac{1}{2}((x_t^{(i)} - x_{t-1}^{(i)})_T)^T \Sigma_{\boldsymbol{\nu}}^{-1} ((x_t^{(i)} - x_{t-1}^{(i)})_T)}}{e^{-(\lambda ||x_t^{(i)} - x_{t-1}^{(i)}||_2^2 + \gamma ||(x_t^{(i)})_{T^c}||_1 + ||y_t - \Phi x_t^{(i)}||_2^2)}} \end{aligned} \tag{5.5}$$

where $T \triangleq N_t^{(i)}$. Details can be found in the Appendix.

5 Estimate $\hat{x}_t = x_t^{(i)}$ where $i = \arg\max_i w_t^{(i)}$

4 Resample [2, 3] on $\{x_t^{(i)}\}_{i=1 \dots N_{PF}}$.

6 Set $t \leftarrow t + 1$ and go to step 2.

CHAPTER 6. Conclusions and Future Directions

In chapter 2, we developed efficient particle filters for solving real-life computer vision problems associated with large dimensional state space and multimodal observation likelihood. In chapter 3, we developed a novel approach to define a generative model for both 2D and 3D nonstationary landmark shape sequences, which we call nonstationary shape activity (NSSA). Applications in filtering, tracking, synthesis (using 3D-NSSA models) and change detection are demonstrated. Filtering and tracking are studied in detail and significantly improved performance over related work is demonstrated. We have also successfully demonstrated the use of the NSSA for model-based compression of landmark shape sequence data. In chapter 4, we have tackled the problem of visual tracking under variable illumination conditions as the inference problem in a joint *motion-illumination* space. We used particle filter with mode tracking (PF-MT) [15] to track on the large dimensional motion-illumination space. This key step ensures accurate tracking with much fewer particles than those needed by any other existing particle filters. Together with this, we also demonstrated the use of change detection algorithm for tracking across abrupt illumination variations. In chapter 6, we have proposed a novel algorithm for sequential estimation (i.e. tracking) of spars signals from a small number of linear measurements. The algorithm utilizes a dynamic prior model on both sparsity pattern change as well as a model on signal dynamics on the known part of the support. In essence, it is a merger between particle filtering and compressive sensing : we call it - Particle Filtered Modified Compressive Sensing(PaFiMoCS). Our simulation experiments lead to promising results for PaFiMoCS as compared to recent state-of-the-art including various compressed sensing based techniques. As we discuss in the next section, this particular approach towards recursive estimation of sparse signals has potential applications for solving real-life

computer vision problems.

6.1 Future Directions

The problems addressed in this thesis and the methods proposed to solve them lead us to several interesting future research directions. In this section, we outline a few directions for future research work.

6.1.1 High Level Vision : Motion Activity Detection, Segmentation and Recognition

In our preceding discussions, we demonstrated how efficient particle filtering together with NSSA based dynamical model can track and detect activity changes from videos. One interesting future direction would be extend this preliminary change detection mechanism for temporal segmentation of motion activities. In other words, we would like to mark the time instants where switching between the motion activities occur and recognize motion additivity class for each segment using state-of-the-art machine learning techniques. There is great potential in using kernel methods for classifying/recognizing motion activities from the time-series shape velocity vectors on the shape space. Also, we could explore several other existing methods and leverage from their ideas. For example, as explained earlier, piecewise ASMs or SSAs are good for recognition problems, but not for automatic tracking, since they do not model the transitions between pieces well. For automatic landmark extraction from image sequences followed by recognition, one could use an NSSA based tracker to extract the landmark sequences, followed by a piecewise ASM or SSA based recognizer as in past work. For change (or abnormality) detection, NSSA can again be successfully used. Another radically different approach towards activity classification/recognition could possibly be the use of neurally inspired models developed in visual object recognition literature(models mimicking the visual cortex of human brain together with statistical classifiers like multi-kernel learning methods, multi-class SVMs [88]). These models can potentially be extended from static(object) to time-series(activity) case. This approach is potentially very promising and there is much to explore.

Another possible future direction towards enhancing the back-end tools of our system is to combine our models with the Gaussian Process Latent Variable Model(GPVLMs) based observation extraction techniques [35, 89]. Also, our optical flow based landmark extractor could be improved by using ideas from [90]. This will make the system more efficient leading us closer to a full-fledged smart computer vision system capable of high level visual understanding like motion activity detection, segmentation and recognition.

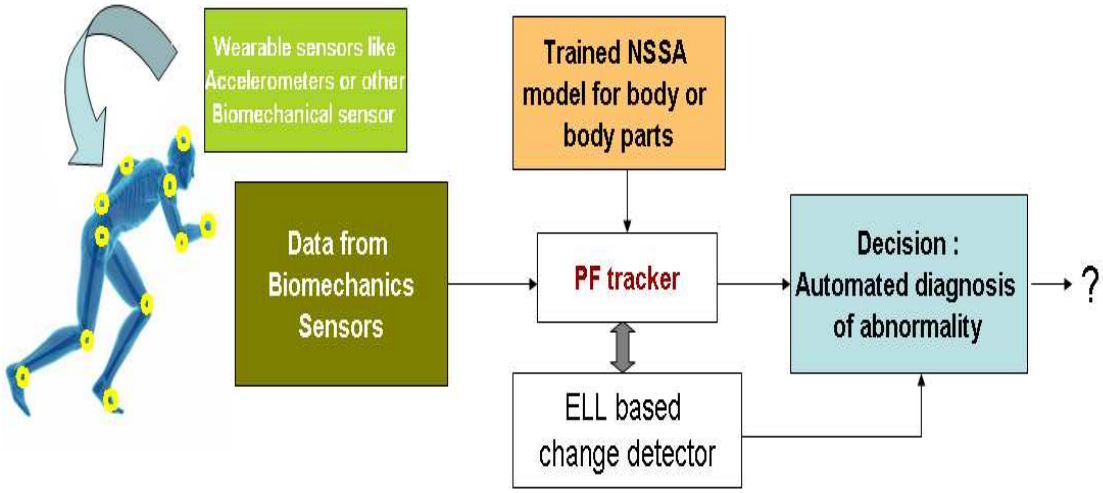


Figure 6.1 An intuitive idea of the overall system for biomechanics applications with automated medical diagnosis. This forms an interesting direction of future research.

6.1.2 Biomechanics : Automated Medical Diagnosis

Another potential future direction would be to extend the dynamical models for landmark shapes towards biomechanics applications for understanding how various body parts interact with each other. In our current research, we use image based landmarks. But there is enormous potential for extending our techniques while using wearable wireless sensors (e.g. accelerometers, location sensor) at various body joints. The time series data from these sensors can be used in a tracking cum abnormality detection framework for automated diagnosis of medical conditions e.g. gait abnormalities due to Parkinson's and Alzheimer's diseases. An intuitive overview of the system is demonstrated in Fig. 6.1. As a part of future research and post-

doctoral work, we would like to take a look at the dynamical models for landmark shapes as a mechanism for learning the temporal dynamics of a group of interacting sensors and develop novel machine learning algorithms enabling medical diagnosis, temporal segmentation and activity recognition from accelerometer data.

6.1.3 PF to PaFiMoCS : Better Visual Tracking

In chapter 4, we developed a dynamical model for illumination using Legendre basis functions. One important limitation of the scheme is the extent of complexity of illumination distribution it can handle. This is because we use basis sets up to a fixed order(5 or less). If we want to incorporate higher order basis functions to facilitate higher modes of illumination variations, it lead us to larger and larger dimension of the state space. This, obviously, is a challenge for any particle filtering framework. One promising direction of future research could be to use modified compressive sensing instead of the mode-tracking step of PF-MT. The advantage of such an approach comes from the fact that even the complex illumination pattern of the scene can have a sparse representation in a Legendre basis set where we deliberately include higher order basis functions up to 20, for example. Now, the illumination of the scene may just need 4 of the functions(of any order up to 20) to be represented with reasonable accuracy. Thus, learning the illumination becomes estimating a 4-sparse vector of length 20. This framework also gives more flexibility on the variation patterns of illumination distribution rather than sticking to a fixed basis set. Here's the basis set contributing to the instantaneous illumination conditions itself could be dynamic and hence more realistic. The estimation of the illumination vector can be formulated from a PaFiMoCS perspective (chapter 5) where we can use a dynamical model on the sparsity pattern change for the illumination vector and utilize regularized modified CS concept due to the fact that illumination variation would be gradual i.e. previous estimate of the illumination vector would be very close the current one. Thus the mode tracking step for illumination particle (in Algorithm 10, chapter 4) can be replace by a PaFiMoCS style step as,

$$\Lambda_t^{(i)} = \arg\min_{\Lambda} \lambda||(\Lambda)_{T^c}||_1 + \gamma||(\Lambda - \Lambda_{t-1}^{(i)})_T||_2^2 + ||Y_t(ROI(U_t^{(i)})) - A\Lambda||_2^2$$

where T is the predicted support and the matrix A incorporates all the Legendre basis functions together with the initial template as describe in chapter 4. The sparse illumination vector Λ is estimated from the above optimization step. It is to be noted that here we are using modified CS even though A is a tall matrix. Basically, we are emphasizing on the sparsity of the underlying signal (prior knowledge) while reconstructing it. Standard least square approach might not result in a sparse solution which does not use the prior knowledge of sparsity. Similar methodology was used in [93] for sparse channel estimation. This framework has another potential application as discussed in the section below.

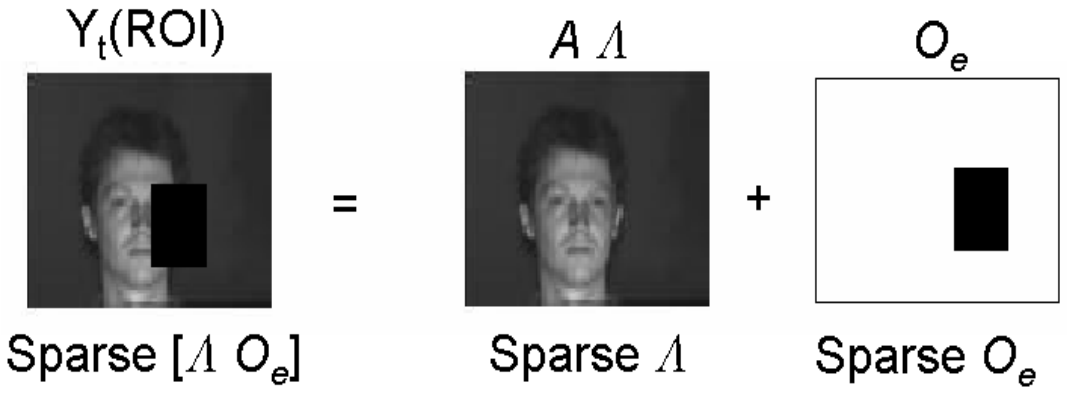


Figure 6.2 Demonstration of the idea for occlusion detection and handling in visual tracking from a compressive sensing perspective. It is inspired by the work of [91] for robust face recognition under occlusion.

6.1.3.1 New Approach to Occlusion Detection and Handling

As mentioned is the section above, the illumination vector can be estimated using compressive sensing style optimization step. What is remarkable is that the same techniques can possibly be very robust to occlusion when the object is partially occluded. As demonstrated in Fig. 6.2, the occlusion can be modeled as a sparse error (denoted as O) similar in spirit to [91, 92]. Now, even with partial occlusion, there is a possibility of robust estimation of both illumination vector and the occlusion error vector. The following problem for estimating the *jointly sparse* vector of illumination parameters and occlusion error can be posed as,

$$[\hat{\Lambda}\ \hat{O}] = \arg\min_{[\Lambda\ O]} ||[\Lambda\ O]||_1 + ||Y_t(ROI) - [A\ I][\Lambda\ O]^T||_2^2 \tag{6.1}$$

It is an approach worth exploring in future work. Now, if the occluding object intensity is different from that of the object, this scheme is expected to reconstruct both illumination vector Λ and occlusion error O. The estimating of O has profound ramification from a computer vision perspective as it not only means occlusion detection but also leads to the localization of occlusion in the current template. Thus we can exactly point out which are the outlier pixels and discard them in the template matching step leading to a more robust visual tracker capable of detecting and handling 'partial occlusion'. Also, in many cases, the occluding object moves slowly over the target. Then, the sparsity pattern of O and in general $[\Lambda\ O]$, would change slowly enabling us to use the tools of modified compressive sensing. One can also use a occlusion motion prediction framework to predict the sparsity pattern change corresponding to O which can again be fit into our PaFiMoCS framework which relies on a dynamical model of the support change (here, considered for the joint sparse vector - $[\Lambda\ O]$). An open question is : given a support estimate T of $[\Lambda\ O]$ can the mod-CS version of the above problem do better ? (in terms of number of measurements required). The corresponding mod-CS version can be posed as,

$$[\hat{\Lambda}\ \hat{O}] = \arg\min_{[\Lambda\ O]} ||[\Lambda\ O]_{T^c}||_1 + ||Y_t(ROI) - [A\ I][\Lambda\ O]^T||_2^2 \tag{6.2}$$

It is to be noted that this idea is still in a conceptual level and would require a lot of analysis, both theoretical and experimental, in order to validate its usefulness. Thus it potentially forms a very interesting direction of future research which can lead to a new approaches towards occlusion detection/handling in visual tracking and benefit computer vision in general.

BIBLIOGRAPHY

[1] G. Welch and G. Bishop, "An introduction to Kalman Filters," *SIGGRAPH*, 2001.

[2] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/nongaussian bayesian state estimation," *IEE Proceedings-F (Radar and Signal Processing)*, pp. 140(2):107–113, 1993.

[3] A. Doucet, "On sequential monte carlo sampling methods for bayesian filtering," in *Technical Report CUED/F-INFENG/TR. 310, Cambridge University Department of Engineering*, 1998.

[4] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Trans. Signal Processing*, vol. 50, pp. 174–188, Feb. 2002.

[5] L. Tierney and J. B. Kadane, "Accurate approximations for posterior moments and marginal densities," vol. 81, no 393, pp. 82–86, March 1986.

[6] T. Schn, F. Gustafsson, and P. Nordlund, "Marginalized particle filters for nonlinear state-space models," *IEEE Trans. Sig. Proc.*, 2005.

[7] D. Kendall, D. Barden, T. Carne, and H. Le, *Shape and Shape Theory.* John Wiley and Sons, 1999.

[8] N. Vaswani, A. RoyChowdhury, and R. Chellappa, ""Shape Activity": A Continuous State HMM for Moving/Deforming Shapes with Application to Abnormal Activity Detection," *IEEE Trans. Image Proc.*, pp. 1603–1616, October 2005.

[9] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[10] CMU-MOCAP, "The carnegie mellon motion capture database cmu graphics lab, http://mocap.cs.cmu.edu,"

[11] S. Das and N. Vaswani, "Model-based compression of nonstationary landmark shape sequences," in *IEEE Intl. Conf. on Image Processing*, 2008.

[12] A. Veeraraghavan, A. RoyChowdhury, and R. Chellappa, "Matching shape sequences in video with an application to human movement analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 12, pp. 1896–1909, 2005.

[13] M. Isard and A. Blake, "Condensation: Conditional Density Propagation for Visual Tracking," *Intl. Journal of Comp. Vision*, pp. 5–28, 1998.

[14] A. Kale and C. Jaynes, "A joint illumination and shape model for visual tracking," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 602–609, 2006.

[15] N. Vaswani, "Particle filtering for large dimensional state spaces with multimodal observation likelihoods," *IEEE Trans. Sig. Proc.*, Accepted (with minor revisions).

[16] N. Vaswani, "Additive change detection in nonlinear systems with unknown change parameters," *IEEE Trans. Sig. Proc.*, pp. 859–872, March 2007.

[17] N. Vaswani and W. Lu, "Modified-cs: Modifying compressive sensing for problems with partially known support," *IEEE Trans. Signal Processing*, September 2010.

[18] N. Vaswani and W. Lu, "Modified-cs: Modifying compressive sensing for problems with partially known support," in *ISIT 2009*, pp. 488 –492, June 2009.

[19] R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan, "The unscented particle filter," in *Advances in Neural Information Processing Systems 13*, Nov 2001.

[20] V. Cevher and J. H. McClellan, "Proposal strategies for joint state-space tracking with particle filters," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2005.

[21] N. Vaswani, "Pf-eis and pf-mt: New particle filtering algorithms for multimodal observation likelihoods and large dimensional state spaces," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[22] N. Vaswani and S. Das, "Particle filter with efficient importance sampling and mode tracking (pf-eis-mt) and its application to landmark shape tracking," in *Asilomar Conf. on Sig. Sys. Comp.*, 2007.

[23] S. Das and N. Vaswani, "Efficient importance sampling techniques for large dimensional and multimodal posterior computations," in *IEEE Digital Signal Processing/SPE Workshop*, 2009.

[24] S. Das and N. Vaswani, "Nonstationary shape activities: Dynamic models for landmark shape change and applications," vol. 32, pp. 579–592, April 2010.

[25] A. Doucet, N. deFreitas, and N. Gordon, eds., *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[26] J. MacCormick and A. Blake, "A probabilistic contour discriminant for object localisation," *IEEE Intl. Conf. on Computer Vision (ICCV)*, Mumbai, India, June 1998.

[27] R. Chen and J. Liu, "Mixture kalman filters," *Journal of the Royal Statistical Society*, vol. 62(3), pp. 493–508, 2000.

[28] I. Dryden and K. Mardia, *Statistical Shape Analysis*. John Wiley and Sons, 1998.

[29] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models: Their training and application," *Computer Vision and Image Understanding*, vol. 61, pp. 38–59, January 1995.

[30] A. Kume, I. Dryden, and H. Le, "Shape space smoothing splines for planar landmark data," *Biometrika*, 2007.

[31] N. Paragios, M. Jolly, M. Taron, and R. Ramaraj, "Active shape models and segmentation of the left ventricle in echocardiography," in *5th Intl. Conf. on Scale Space and PDE Methods in Computer Vision, Hofgeismar, Germany*, April 2005.

[32] B. Song, N. Vaswani, and A. K. Roy-Chowdhury, "Closedloop tracking and change detection in multi-activity sequences," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[33] T.F.Cootes, G. Edwards, and C.J.Taylor, "Active appearance models," in *European Conf. on Computer Vision*, vol. 2, pp. 484–498, 1998.

[34] B. P. Lelieveldt, R. J. van der Geest, J. H. C. Reiber, J. G. Bosch, S. C. Mitchel, and M. Sonka, "Time-continuous segmentation of cardiac image sequences using active appearance motion models," *IPMI*, pp. 446–452, January 2001.

[35] T. Tai-Peng, L. Rui, and S. Sclaroff, "Tracking human body on a learned smooth space," in *Boston University Computer Science Tech. Report No. 2005-029*, 2005.

[36] S. Hou, A. Gatala, F. Caillette, N. Thacker, and P. Bromiley, "Real-time body tracking using a gaussian process latent variable model," in *IEEE Intl. Conf. on Computer Vision (ICCV)*, October 2007.

[37] A. Kume, I. Dryden, H. L. Le, and A. Wood, "Fitting cubic splines to data in shape spaces of planar configurations," in *Leeds Annual Statistics Research*, 2002.

[38] A. Srivastava and E. Klassen, "Bayesian and geometric subspace tracking," *Advances in Applied Probability*, pp. 43–56, March 2004.

[39] S. X. Ju., M. J. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, 1996.

[40] N. Vaswani, "Additive change detection in nonlinear systems with unknown change parameters," *IEEE Trans. Sig. Proc.*, 2006, accepted.

[41] N. Vaswani and R. Chellappa, "Nonstationary shape activities," in *IEEE Conf. Decision and Control (CDC)*, 2005.

[42] P. Yan and A. Kassim, "Lossless and near-lossless motion-compensated 4d medical image compression," in *IEEE International Workshop On Biomedical Circuits And Systems*, 2004.

[43] S. Sclaroff and A.P.Pentland, "On modal modeling for medical images: Underconstrained shape description and data compression," in *Biomedical Image Analysis*, June 1994.

[44] J. L. Su, C. C. Lin, J. R. Duann, and Y. S. Tsai, "Biomedical image compression using vector quantization algorithm," in *IEEE International Conference on Engineering in Medical and Biology Society*, pp. 66–67, Oct 1993.

[45] N. Brady, F. Bossen, and N. Murphy, "Contex-based arithmatic encoding of 2d shape sequences," in *IEEE Intl. Conf. on Image Processing*, pp. 26–29, Oct 1997.

[46] H. Zang and F. Bossen, "Region-based coding of motion fields for low bit-rate video compression," in *IEEE Intl. Conf. on Image Processing*, pp. 24–27, Oct 2004.

[47] S. Das, A. Kale, and N. Vaswani, "Particle filter with mode tracker (pf-mt) for visual tracking across illumination changes," *IEEE Trans. Image Proc.*, p. (under review).

[48] Y. Xu and A. K. Roy-Chowdhury, "Integrating motion, illumination, and structure in video sequences with applications in illumination-invariant tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, 2007.

[49] J. R. Basri and D. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 2, pp. 218–233, 2003.

[50] R. Ramamoorthi, "Analytic pca construction for theoretical analysis of lighting variability in images of lambertian object," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 10, pp. 1–12, 2002.

[51] B. P. Belhumeur and D. J. Kriegman, "What is the set of images of an object under all possible illumination conditions," vol. 28, no. 3, pp. 1–16, 1998.

[52] G. Hager and P. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 10, p. 10251039, 1998.

[53] P. Belhumeur and D.J.Kriegman, "What is the set of images of an object under all possible illumination conditions," *Intl. Journal of Comp. Vision*, vol. 28, no. 3, pp. 1–16, 1998.

[54] B.Han and L. Davis, "On-line density-based appearance modeling for object tracking," in *IEEE Intl. Conf. on Computer Vision (ICCV)*, 2005.

[55] S. Zhou, R. Chellappa, and B. Moghaddam, "Visual tracking and recognition using appearance-adaptive models in particle filters," *IEEE Trans. Image Proc.*, November 2004.

[56] A.D.Jepson, D.J.Fleet, and T. Maraghi, "Robust online appearance models for visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, October 2003.

[57] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," in *British Machine Vision Conference*, September 2003.

[58] D. Comaniciu, V. Ramesh, and P. Meer, "Real time tracking of non-rigid objects using mean shift," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[59] D. A. Ross, J. Lim, Ruei-Sung, and L. M.-H. Yang, "Incremental learning for robust visual tracking," *Intl. Journal of Comp. Vision*, vol. 77, pp. 125–141, 2008.

[60] J. Jackson, A. Yezzi, and S. Soatto, "Dynamic shape and appearance modeling via moving and deforming layers," *Intl. Journal of Comp. Vision*, vol. 79, pp. 71–84, August 2008.

[61] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1728–1740, 2008.

[62] A. Sung, T. Kanade, and D. Kim, "Pose robust face tracking by combining active appearance models and cylinder head models," *Intl. Journal of Comp. Vision*, vol. 80, no. 2, pp. 260–274, 2008.

[63] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[64] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 6, pp. 681–685, 2001.

[65] S. Das and N. Vaswani, "Particle filtered modified-cs (pafimocs)," 2010.

[66] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 53(12), pp. 4655–4666, December 2007.

[67] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. on Information Theory*, (Submitted) 2007.

[68] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," (Preprint) 2008.

[69] S. S. Chen, D. L. Donoho, Michael, and A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[70] E. Candes and T. Tao, "Near optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. on Information Theory*, vol. 52(12), pp. 5406 – 5425, December 2006.

[71] D. Donoho, "Compressed sensing," *IEEE Trans. on Information Theory*, vol. 52(4), pp. 1289–1306, April 2006.

[72] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Info. Theory*, vol. 52(2), pp. 489–509, February 2006.

[73] N. Vaswani, "Kalman filtered compressed sensing," in *IEEE Intl. Conf. on Image Processing*, 2008, submitted.

[74] D. Angelosante, E. Grossi, and G. B. Giannakis, "Compressed sensing of time-varying signals," in *DSP*, 2009.

[75] D. Angelosante,G. B. Giannakis,"RLS-weighted Lasso for adaptive estimation of sparse signals" in *IEEE Trans. on Signal Proc.*

[76] M. A. Khajehnejad, W. Xu, A. S. Avestimehr, and B. Hassibi, "Weighted l1 minimization for sparse recovery with prior information," in *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 1*, ISIT'09, pp. 483–487, 2009.

[77] P. Schniter, L. C. Potter, and J. Ziniel, "Fast bayesian matching pursuit: Model uncertainty and parameter estimation for sparse linear models," 2009.

[78] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," 2007.

[79] J. Kent, "The complex bingham distribution and shape analysis," in *Journal of the Royal Statistical Society, Series B*, pp. 56:285–299, 1994.

[80] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[81] S. Khan, "Matlab code for optical flow using lucas kanade method," *www.cs.ucf.edu/ khan/.*

[82] T. Cover and J. Thomas, *Elements of Information Theory.* Wiley Series, 1991.

[83] S. Das, S. Rane, and A. Vetro, "Information hiding inside structured shapes," *IEEE Transactions on Information Forensics and Secutiry (to be submitted).*

[84] S. Das, S. Rane, and A. Vetro, "Information hiding inside structured shapes," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2010.

[85] T. Schn, F. Gustafsson, and P. Nordlund, "Marginalized particle filters for nonlinear state-space models," *IEEE Trans. Sig. Proc.*, 2005.

[86] D. Kerridge, "Inaccuracy and inference," *J. Royal Statist. Society, Ser. B*, vol. 23 1961.

[87] M. Pitt and N. Shephard, "Filtering via simulation: auxiliary particle filters," *J. Amer. Stat. Assoc*, vol. 94, p. 590599, 1999.

[88] N. Pinto, J. DiCarlo, and D. Cox, "How far can you get with a modern face recognition test set using only simple features?," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 2591–2598, 2009.

[89] N. Lawrence, "Probabilistic non-linear principal component analysis with gaussian process latent variable models," *Journal of Machine Learning Research*, pp. 1783–1816, November 2005.

[90] A. Srivastava and H. Jermyn, "Looking for shapes in two-dimensional cluttered point clouds," *IEEE Trans. Pattern Anal. Machine Intell.*, to appear.

[91] J. Wright, A. Ganesh, A. Yang and Y. Ma,"Robust face recognition via sparse representation" *IEEE Trans. Pattern Anal. Machine Intell.*, 2008.

[92] J. Wright and Y. Ma, "Dense error correction via l1 minimization" *Arxiv.org* , 2008.

[93] S. Cotter and B. Rao, "Sparse Channel Estimation via Matching Pursuit With Application to Equalization" *IEEE Trans. on Communications*, vol. 50(3), March 2002.