Learning Probabilistic Models for Visual Motion

by

David Alexander Ross

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Learning Probabilistic Models for Visual Motion

David Alexander Ross

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2008

A fundamental goal of computer vision is the ability to analyze motion. This can range from the simple task of locating or tracking a single rigid object as it moves across an image plane, to recovering the full pose parameters of a collection of nonrigid objects interacting in a scene. The current state of computer vision research, as with the preponderance of challenges that comprise "artificial intelligence", is that the abilities of humans can only be matched in very narrow domains by carefully and specifically engineered systems.

The key to broadening the applicability of these successful systems is to imbue them with the flexibility to handle new inputs, and to adapt automatically without the manual intervention of human engineers. In this research we attempt to address this challenge by proposing solutions to motion analysis tasks that are based on machine learning.

We begin by addressing the challenge of tracking a rigid object in video, presenting two complementary approaches. First we explore the problem of learning a particular choice of appearance model—principal components analysis (PCA)—from a very limited set of training data. However, PCA is far from the only appearance model available. This raises the question: given a new tracking task, how should one select the most-appropriate models of appearance and dynamics? Our second approach proposes a data-driven solution to this problem, allowing the choice of models, along with their parameters, to be learned from a labelled video sequence.

Next we consider motion analysis at a higher-level of organization. Given a set of trajectories obtained by tracking various feature points, how can we discover the underlying non-rigid structure of the object or objects? We propose a solution that models the observed sequence in terms of probabilistic "stick figures", under the assumption that the relative joint angles between sticks can change over time, but their lengths and connectivities are fixed. We demonstrate the ability to recover the invariant structure and the pose of articulated objects from a number of challenging datasets.

Acknowledgements

This thesis could not have been completed without the invaluable contributions of my advisor Richard Zemel, and my collaborators Jongwoo Lim, Ruei-Sung Lin, Edward Meeds, Simon Osindero, Sam Roweis, Daniel Tarlow, and Ming-Hsuan Yang.

Additional assistance with experiments and figures was provided by Thomas El-Maraghi, Bohyung Han, Allan Jepson, Volodymyr Mnih, Fabian Wauthier, and Karolina Büchner.

The motion capture data used in this project was provided by the Biomotion Lab, Queen's University, Canada, and the Carnegie Mellon University Motion Capture Database http://mocap.cs.cmu.edu/ (created with funding from NSF EIA-0196217).

Work presented in this thesis was supported in part by the Government of Canada, through a Canada Graduate Scholarship, and by an internship at Honda Research Institute, USA.

Contents

Chapter 1

Introduction

Computer vision considers the challenge of creating programs which process sensory input—in this case patterns of light acquired by cameras—and from from it develop a semantic understanding of the visible world. As it provides a mechanism for translating the physical to the conceptual, computer vision falls clearly in the field of artificial intelligence: building working models with capabilities equalling or surpassing those of the human mind.

A fundamental component of computer vision is the ability to analyze motion. This can range from the simple task of locating or tracking a single rigid object as it moves across an image plane, to recovering the full pose parameters of a collection of nonrigid objects interacting in a scene.

The current state of computer vision research, as with all challenges that comprise the field of artificial intelligence, is that the abilities of humans can only be matched in very narrow domains by carefully and specifically engineered systems. For example, numerous programs exist which can track a human face as it moves in a video. However many of these have difficulty coping with dramatic changes in lighting, appearance and pose, or handling partial or total occlusions of the target. At this point there exists no universal solution capable of dealing with the full range of variability to which humans

are accustomed.

Similarly, systems exist which can recover, to some extent, the full three-dimensional pose of a human body from video as it moves. However these predominantly employ a mathematical model of the human body, describing its configuration and range of motion, which must be painstakingly constructed by experts. The requirement of such a model limits the applicability of the system to new but related challenges, for example recovery of the 3D pose of a giraffe.

The key to broadening the applicability of these successful systems is to imbue them with the flexibility to handle new inputs, and to adapt automatically without the manual intervention of human engineers. In this research we attempt to address this challenge by proposing solutions to the aforementioned motion analysis tasks that are based on machine learning.

Machine learning is the study of programs which adapt their behaviour and performance based on the analysis of observed data, such as instances of the task that needs to be performed (*e.g.* images of the face that is to be tracked). Our focus in this work will be on probabilistic learning techniques, which fit statistical models to the data, then solve the task by performing inferences using the model. The advantage of the probabilistic approach is that it provides a principled way to account for noise in the observations and uncertainty in the state of the world. The learning techniques appearing in this work are varied; we extend existing methods where applicable, and develop new approaches when necessary.

1.1 Visual Tracking

The first problem we address is the challenge of tracking a rigid object in video. Using the popular probabilistic *state-space model* framework, tracking is commonly divided into modelling the target's appearance, allowing it to be detected in each video frame,

and modelling its dynamics, enabling predictions to be made regarding its motion in subsequent frames. Thus, creating a tracker involves selecting specific types of appearance and dynamics models, as well as fitting these models to the available data.

The first approach we explore deals with the problem of learning a particular choice of appearance model—principal components analysis (PCA)—from a very limited set of training data. However, PCA is far from the only appearance model available. This raises the question: given a new tracking task, how should one select the most appropriate appearance and dynamics models? Our second approach proposes a data-driven solution to this problem, allowing the choice of models, along with their parameters, to be learned from a labelled video sequence.

1.1.1 Incremental Learning of Appearance

PCA is a widely used appearance model, due to its ability to cope with changes in lighting, as well as other variability in the appearance of the target. To fit a PCA model, training images depicting the full range of possible variation (or as much as possible) are required, and therefore detailed knowledge of the target's appearance must be available before tracking begins. However, often when tracking, the only information available is the target's location, and thus appearance, in the first frame of the video. To deal with this, we propose to learn the PCA model incrementally, online and efficiently, while tracking. This works has been presented in (Ross et al., 2004), (Lim et al., 2005b), and (Ross et al., 2008a).

The contributions of this project include:

- A new incremental algorithm for learning PCA that is exact, correctly updates the subspace's mean, is efficient in terms of time and space (update requirements are constant, irrespective of the number of observations seen so far), faster than its competitors, and includes an optional "forgetting factor" that leads to measurable improvements in tracking performance.

- A very short proof that illustrates, when the forgetting factor is used, the objective function that is being optimized.
- A tracking algorithm that, by incorporating incremental PCA, allows the popular eigentracking approach to be used when a dataset of training images is not available prior to tracking.
- Qualitative and quantitative experimental results that the resulting tracker is competitive with the state of the art.

1.1.2 Tracking with Learned Combinations of Discriminative Features

For our second approach we propose a new model for the probabilistic estimation of continuous state variables from a sequence of observations, and demonstrate how it can be applied to tracking the position of an object in video. This mapping is modeled as a product of dynamics experts (features relating the state at adjacent time-steps) and observation experts (features relating the state to the image sequence). Individual features are flexible in that they can switch on or off at each time-step depending on their inferred relevance (or on additional side information), and discriminative in that they need not model the full generative likelihood of the data. When trained conditionally, this permits the inclusion of a broad range of rich features (for example, features relying on observations from multiple time-steps), and allows the relevance of features to be learned from labeled sequences. This approach was initially presented in (Ross et al., 2006).

The contributions of this project include:

- A new discriminative learning model for continuous vector-valued sequences. This provides a novel way to apply the ideas behind the Conditional Random Field to continuous variables, and Products of Experts to conditional probabilistic models.

- The ability to learn which dynamics and appearance features (models) are appropriate for a given task, and to switch features on and off dynamically based on their estimated reliability. The result is a tracker that is more reliable than any of its constituent appearance and dynamics models.

- A tracking algorithm which is able to track through total occlusions of the target.

1.2 Learning Articulated Structure and Motion

Although the ability to track a single rigid object serves as a key basic operation, in practice many interesting objects are not rigid. For example, humans can be more accurately modelled as articulated skeletons, consisting of a number of rigid body parts connected by joints. Performing motion analysis at this higher level of organization provides a number of advantages. First, connectivity allows the location of a missing part to be inferred even when it cannot be seen, permitting pose estimates to be made robustly, even in presence of partial occlusion and tracking failures. Moreover, sequences of actions are often more easily described by the angles between adjoining body parts, thereby facilitating activity recognition and other related tasks. Recognizing this, the second problem we address is the recovery of three-dimensional skeletal structure, from the observed locations of a set of feature points tracked in the 2D image plane. This consists of simultaneously learning the time invariant structure (grouping of features into rigid parts, and the connectivity between parts), and estimating the pose (motion parameters or joint angles) of the skeleton in each frame.

We model the observed sequence in terms of probabilistic "stick figure" objects, under the assumption that the relative joint angles between sticks can change over time, but their lengths and connectivities are fixed. We formulate the problem in a single probabilistic model that includes multiple sub-components: associating the features with particular sticks, determining the proper number of sticks, and finding which sticks are

physically joined. The model is fit to the observations using a combination of greedy learning and resampling for the structure, and expectation-maximization for the remaining parameters. We test the algorithm on challenging datasets of 2D projections of optical human motion capture and feature trajectories from real videos. This work has been presented in (Ross et al., 2007) and (Ross et al., 2008b), and is a follow-up to Multiple Cause Factor Analysis (Ross and Zemel, 2006).

The contribution of this project is a robust algorithm for recovering skeletons from feature point trajectories, with a number of advantages over existing state-of-the-art approaches including:

- The ability to recover 3D structure from 2D feature trajectories, on a variety of challenging datasets, and in the presence of occluded training data.
- An explicit (joint probability) objective function that is optimized.
- A quantitative method for evaluating performance, based on predicting the locations of held-out observations, in the presence of realistic occlusions.
- The ability to trivially extend the algorithm to the case of 3D observations as well.

1.3 Organization

The remainder of the thesis is organized around three main chapters, one for each of the aforementioned projects, followed by a discussion and supplemental material. Chapter 2, *Incremental Learning for Visual Tracking*, begins with work on tracking using an incrementally trained PCA appearance model, Chapter 3 covers the work on *Combining Discriminative Features to Infer Complex Trajectories*, and Chapter 4 on *Learning Articulated Structure and Motion*. Concluding remarks and suggestions for future research are presented in Chapter 5.

The main text of this thesis is supported by three appendices and a collection of videos displaying additional experimental results. The first appendix, Appendix A, describes the Multivariate Gaussian probability distribution, presenting identities that are employed a number of times in this work. The second, Appendix B, gives a detailed derivation of inference for the *Combining Discriminative Features* model. Finally, Appendix C derives the updates used by the expectation-maximization algorithm in Chapter 4. The accompanying videos, as well as Matlab source code for all of the projects presented in this work, are available online at http://www.cs.toronto.edu/~dross/phd/.

Chapter 2

Incremental Learning for Robust Visual Tracking

Visual tracking is concerned with locating and following a target object in a constantly changing stream of images. While most existing algorithms are able to track objects well in controlled environments, they usually fail in the presence of significant variation of the object's appearance or surrounding illumination. One reason for such failures is that many algorithms employ fixed appearance models of the target. Such models are trained using only appearance data available before tracking begins, which in practice limits the range of appearances that are modelled, and ignores the large volume of information (such as shape changes or specific lighting conditions) that becomes available during tracking. In this chapter, we present a tracking method that incrementally learns a low-dimensional subspace representation, efficiently adapting online to changes in the appearance of the target. The model update, based on incremental algorithms for principal component analysis, includes two important features: a method for correctly updating the sample mean, and a forgetting factor to ensure less modelling power is expended fitting older observations. Both of these features contribute measurably to improving overall tracking performance. Numerous experiments demonstrate the effectiveness of the proposed

tracking algorithm in indoor and outdoor environments where the target objects undergo large changes in pose, scale, and illumination.

2.1 Introduction

Visual tracking essentially deals with non-stationary data, as the appearances of both the target object and the background change over time. Most existing algorithms are able to track objects, either previously viewed or not, in short durations and in well controlled environments. However these algorithms usually fail to observe the object motion or have significant drift after some period of time, due to drastic change in the object's appearance or large lighting variation in its surroundings. Although such situations can be ameliorated with recourse to richer representations, effective prediction schemes or combination, most algorithms typically operate on the premise that the model of the target object does not change drastically over time. Examples abound, ranging from representation methods based on view-based appearance models (Black and Jepson, 1996), contours (Isard and Blake, 1996), parametric templates of geometry and illumination (Hager and Belhumeur, 1996), integration of shape and color (Birchfield, 1998), mixture models (Black et al., 1998), 3D models (La Cascia and Sclaroff, 1999), exemplars (Toyama and Blake, 2001), foreground/background models (Harville, 2002), templates with updating (Matthews et al., 2004), prediction schemes using particle filters (Isard and Blake, 1996), joint probabilistic data association filters (Rasmussen and Hager, 1998), kernel-based filters (Comaniciu et al., 2003; Georgescu et al., 2004), support vector machines (Avidan, 2001; Williams et al., 2003) and variational inference (Vermaak et al., 2003). These algorithms usually build or learn a model of the target object first and then use it for tracking, without adapting the model to account for changes in the appearance of the object (*e.g.* large variation of pose or facial expression) or the surroundings (*e.g.* lighting variation). Furthermore, it is assumed that all images are acquired with a

stationary camera. Such an approach, in our experience, leads to a brittle tracker that works well only under carefully controlled conditions.

The chief challenge of visual tracking can be attributed to the difficulty in handling the appearance variability of a target object. Intrinsic appearance variability includes pose variation and shape deformation,whereas extrinsic illumination change, camera motion, camera viewpoint, and occlusions inevitably cause large appearance variation. Due to the nature of the tracking problem, it is imperative for a robust algorithm to model such appearance variation.

In this chapter we propose a method that, during visual tracking, efficiently learns and updates a low dimensional subspace representation of the target object. The advantages of this adaptive subspace representation are several fold. The subspace representation provides a compact notion of the "thing" being tracked rather than treating the target as a set of independent pixels, *i.e.* "stuff" (Adelson and Bergen, 1991), and facilitates object recognition. An efficient incremental method continually updates the subspace model to reflect changes in appearance caused by intrinsic and extrinsic factors, thereby facilitating the tracking process. Incrementally updating the subspace removes the offline learning phase required by other eigentrackers, allowing one to track objects for which a database of training images is not even available. To estimate the locations of the target objects in consecutive frames, we use a sampling algorithm with likelihood estimates, which is in contrast to other tracking methods that usually solve complex optimization problems using gradient descent. Furthermore, while numerous algorithms operate under the assumption that there there is no camera motion, our method is able to track objects without this constraint.

The remaining part of this chapter is organized as follows. We begin, in the next section, by reviewing the most relevant algorithms that motivated this work. The details of our algorithm are described in Section 2.3, where we propose an efficient incremental subspace method with a mean update and forgetting factor, followed by an effective

tracking algorithm. The results of numerous experiments and performance evaluation are presented in Section 2.4. We conclude this chapter with remarks on potential extensions for future work. The data, source code, and videos corresponding to this work can all be found at http://www.cs.toronto.edu/~dross/phd/.

2.2 Related Work and Motivation

There is a rich literature in visual tracking, and writing a comprehensive survey on this topic would be a daunting project in and of itself. In this section we review only the most relevant visual tracking work, focusing on algorithms that operate directly on grayscale images. We contrast our method with these methods in terms of their representation scheme, target prediction approach, and their ability to handle changes in illumination as well as appearance.

Visual tracking problems have conventionally been formulated as prediction tasks within which fixed templates and optical flow techniques are utilized to estimate the motion of a target object (Lucas and Kanade, 1981). Such approaches do not take the appearance variability into consideration, and thus perform well only over short periods of time. To enhance the robustness of such object trackers, Black and Jepson proposed an algorithm using a pre-trained view-based eigenbasis representation and a robust error norm (Black and Jepson, 1996). Instead of relying on the brightness constancy principal assumed in optical flow techniques, they advocated the use of a subspace constancy assumption for motion estimation. Although their algorithm demonstrated excellent empirical results, it entailed learning a set of view-based eigenbases before the tracking task began. To achieve robust visual tracking with this method, it is imperative to collect a large set of training images covering the range of possible appearance variation (including viewing angles and illumination) from which to construct the eigenbasis, as this representation, once learned, is not updated.

Observing that low-dimensional linear subspaces are able to effectively model image variation due to illumination (Belhumeur and Kreigman, 1997), Hager and Belhumeur developed a tracking algorithm to handle the appearance variation caused by illumination and pose change using parametric models (Hager and Belhumeur, 1996). Their method extends a gradient-based optical flow algorithm by incorporating low-dimensional representations (Belhumeur and Kreigman, 1997) for object tracking under varying illumination conditions. Before tracking begins, a set of illumination bases needs to be constructed at a fixed pose in order to account for changes in appearance due to lighting variation. However, this basis does not attempt to account for changes in pose such as out-of-plane rotations.

Realizing the limitations of having a single (unimodal or Gaussian) hypothesis of target location at each time-step—as produced by the Kalman filter and its relatives—Isard and Blake introduced particle filters to visual tracking and presented the Condensation algorithm for contour tracking in which multiple plausible interpretations are propagated over time (Isard and Blake, 1996). This probabilistic approach has demonstrated success in tracking the outline of target objects in clutter. However, the representation scheme employed (curves or splines) ignores the internal appearance of the target, and is not updated to account for variations in its appearance, due to pose or illumination change. Instead, Condensation can be paired with alternative appearance models. For example Khan et al. (2004) have combined a variant of Condensation (Rao-Blackwellized particle filtering) with an eigentracker, significantly improving the performance of eigentracking in cluttered scenes.

Supervised discriminative methods for classification and regression have also been exploited to solve visual tracking problems. For example, Avidan (2001) developed a tracking algorithm that employs the support vector machine (SVM) classifier within a optic flow framework (Avidan, 2001). Avidan modified the conventional use of the SVM classification score to instead predict target location, by computing image gradients as

is done in optical flow algorithms. Although this algorithm has demonstrated success in tracking specific objects, such as cars from a mounted camera in a moving vehicle, significant effort is required in training a SVM. Along similar lines, Williams et al. developed a method in which an SVM-based regressor was used for tracking (Williams et al., 2003). Instead of relying on optical flow to predict object location, they learned a perturbation function of spatial in-plane displacements between frames, thereby predicting the most likely object location. As a result of training the regressor on in-plane image motion, this method is not effective in tracking objects with out-of-plane movements.

Mixture models have been studied as alternatives to linear representations, to better account for appearance change in motion estimation. Black et al. (1998) identified four possible factors causing appearance change, fitting them with a mixture model which was then used to estimate image motion. A more elaborate mixture model fit via an online EM algorithm was recently proposed by Jepson et al. (2003), in which three components were used to model the responses of wavelet filters, and thereby account for appearance variation during tracking. Their method is able to handle variations in pose, illumination and expression. However, their appearance model treats pixels within the target region independently (ignoring their covariance) and thus does not have notion of the "thing" being tracked. This can result in modelling background rather than the foreground, thereby failing to track the target object(Jepson et al., 2003).

Attempts to improve the classic Lucas-Kanade tracker (Lucas and Kanade, 1981) with updates was recently made by Matthews et al. (2004). They developed a template update method for visual tracking, which employs an active appearance model (Cootes et al., 2001) to account for image variation. Thus instead of using a fixed template, the object appearance is modelled by a linear combination of appearance images. The tracking problem is then formulated as a search (using gradient descent) for the affine parameters and linear combination which minimize the difference between the target object and the current appearance model. The newly tracked object is then used to update appearance

model, as necessary. They demonstrated good tracking results on vehicles and faces with varying expressions. However, the authors noted that the computation cost for updating the template increases dramatically as principal component analysis is carried out at each update, and that their work covers the case where the visibility of the target object does not change.

Our work is motivated in part by the prowess of subspace representations as appearance models (Murase and Nayar, 1995; Belhumeur and Kreigman, 1997), the effectiveness of particle filters (Isard and Blake, 1996), and the adaptability of online update schemes (Jepson et al., 2003). In contrast to the eigentracking algorithm (Black and Jepson, 1996), our algorithm does not require a training phase but learns the eigenbases online during the object tracking process. Thus our appearance model can adapt to changes in pose, view angle, and illumination not captured by the set of training images—in fact the need to manually collect training images prior to tracking is eliminated. Further, our method uses a particle filter for motion parameter estimation rather than the gradient descent method, which often gets stuck in local minima or is distracted by outliers (Black and Jepson, 1996). Our appearance-based model provides a richer description than simple curves or splines as used in (Isard and Blake, 1996), and has a stronger notion of the "thing" being tracked. In addition, the learned representation can be utilized for other tasks such as object recognition. With respect to the template update method (Cootes et al., 2001), we concurrently developed an efficient subspace update algorithm that facilitates object tracking under varying pose and lighting conditions. Furthermore, our algorithm is able to handle camera motion while learning compact representations and tracking objects. In this work, an eigenbasis representation is learned directly from pixel values corresponding to a target object in the image space. Experiments show that good tracking results can be obtained using this representation without employing more complicated wavelet features as in (Jepson et al., 2003), although this elaboration is still possible and may lead to even better results. Note also that the view-based eigenbasis

representation has demonstrated its ability to model the appearance of objects in different poses (Murase and Nayar, 1995), and under different lighting conditions (Belhumeur and Kreigman, 1997). Consequently, the learned eigenbasis facilitates tracking objects undergoing illumination and pose change.

2.3 Incremental Learning for Tracking

We present details of the proposed incremental learning algorithm for object tracking in this section. First we propose an efficient method that incrementally updates an eigenbasis as new observations arrive, which is used to learn the appearance of the target while tracking progresses. Next we describe our approach for drawing particles in the motion parameter space and predicting the most likely object location with the help of the learned appearance model. Collectively, we show how these two modules work in tandem to track objects well under varying conditions.

2.3.1 Incremental Update of Eigenbasis and Mean

The appearance of a target object may change drastically due to intrinsic and extrinsic factors as discussed earlier. Therefore, to produce a robust tracker, it is important to adapt the appearance model online, while tracking, to reflect these changes. The appearance model we have chosen, a eigenbasis, is typically learned offline from a set of training images $\{I_1, \ldots, I_n\}$, by taking computing the eigenvectors U of the sample covariance matrix $\frac{1}{n-1}\sum_{i=1}^{n}(I_i - \bar{I})(I_i - \bar{I})^\top$, where $\bar{I} = \frac{1}{n}\sum_{i=1}^{n} I_i$ is the sample mean of the training images. Equivalently one can obtain U by computing the singular value decomposition $U\Sigma V^\top$ of the centered data matrix $[(I_1 - \bar{I}) \ldots (I_n - \bar{I})]$, with columns equal to the respective training images minus their mean.

Adapting the appearance model to account for novel views of the target can be thought of as retraining the eigenbasis with an additional m images $\{I_{n+1}, \ldots, I_{n+m}\}$, for

some value of m. Naively, this update could be performed by computing the singular value decomposition $U'\Sigma'V'^{\top}$ of the augmented (centered) data matrix $[(I_1 - \bar{I}') \ldots (I_{n+m} - \bar{I}')]$, where $\bar{I}'$ is the average of the entire $n+m$ training images.

Unfortunately this approach is unsatisfactory for online applications, like visual tracking, due to its storage and computational requirements. First, the naive approach uses the entire set of training images for each update. If an update is made at each video frame, then the number of images which must be retained grows linearly with the length of the sequence. Second, the cost of computing the mean and singular value decomposition grows with the number of images, so the algorithm will run ever slower as time progresses. Instead, the requirements of our application dictate that any algorithm for updating the mean and eigenbasis must have storage and computational requirements that are constant, regardless of the number of images seen so far.

Numerous, more-sophisticated algorithms have been developed to efficiently update an eigenbasis as more data arrive (Golub and Van Loan, 1996) (Hall et al., 1998) (Levy and Lindenbaum, 2000) (Brand, 2002). However, most methods assume the sample mean is fixed when updating the eigenbasis, or equivalently that the data is inherently zero-mean. Neither assumption is appropriate in our application. An exception is the method by Hall et al. (2002), which does consider the change of the mean as each new datum arrives. Although similar to our (independently developed) algorithm, it lacks the forgetting factor, which hurts its suitability for tracking, and has a greater computational cost. (Both of these disadvantages are demonstrated quantitatively in Section 2.4.3.) Hall's algorithm is based on the notion of adding eigenspaces. As a result, some of the additional complexity comes from computing the eigenvalue decomposition of each block of new data as it arrives. In this respect our algorithm is simpler, since it incorporates new data directly, without the additional step.

Here we extend one of these efficient update procedures—the Sequential Karhunen-Loeve (SKL) algorithm of Levy and Lindenbaum (2000)—presenting a new incremental

PCA algorithm that correctly updates the eigenbasis as well as the mean, given one or more additional training data. Our algorithm, a variation of which was first presented in (Lim et al., 2005a), has also been applied to algorithms where the subspace mean plays an important role. For example, it can be applied to adaptively update the between-class and within-class covariance matrices used in Fisher linear discriminant analysis (Lin et al., 2005). We begin with a summary of the SKL algorithm, then describe our new incremental PCA algorithm, and follow with a discussion of a forgetting factor which can be used to down-weight the effect of earlier observations on the PCA model.

Putting aside for the moment the problem of the sample mean, suppose we have a $d \times n$ data matrix $A = \{\mathbf{I}_1, \ldots, \mathbf{I}_n\}$ where each column $\mathbf{I}_i$ is an observation (a d-dimensional image vector in this chapter), for which we have already computed the singular value decomposition $A = U\Sigma V^\top$. When a $d \times m$ matrix B of new observations is available, the goal is to efficiently compute the SVD of the concatenation of A and B: $[A\ B] = U'\Sigma' V'^\top$. Letting $\tilde{B}$ be the component[1] of B orthogonal to U, we can express the concatenation of A and B in a partitioned form as follows:

$$\begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} U & \tilde{B} \end{bmatrix} \begin{bmatrix} \Sigma & U^\top B \\ 0 & \tilde{B}^\top B \end{bmatrix} \begin{bmatrix} V^\top & 0 \\ 0 & I \end{bmatrix}. \tag{2.1}$$

Let $R = \left[\begin{smallmatrix} \Sigma & U^T B \\ 0 & \tilde{B}^\top B \end{smallmatrix}\right]$, which is a square matrix of size $k + m$, where k is the number of singular values in Σ. The time required to compute the SVD of R, $R = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$, does not depend on n, the number of data in A. Now the SVD of $[A\ B]$ can be expressed as

$$\begin{bmatrix} A & B \end{bmatrix} = \left(\begin{bmatrix} U & \tilde{B} \end{bmatrix} \tilde{U} \right) \tilde{\Sigma} \left(\tilde{V}^\top \begin{bmatrix} V^\top & 0 \\ 0 & I \end{bmatrix} \right).$$

Since an incremental PCA is only interested in computing U' and Σ', V', whose size scales with the number of observed data, need not be computed. Thus we arrive at the following formulation of the SKL algorithm (Levy and Lindenbaum, 2000).

[1] More precisely, $\tilde{B}$ is a matrix with orthonormal columns spanning $range(B) \setminus range(A)$.

Given U and Σ from the SVD of A, compute U' and Σ' from the SVD of $[A\ B]$:

1. Obtain $\tilde{B}$ and R by taking the QR decomposition of $[U\Sigma\ B]$: $[U\ \ \tilde{B}]R \stackrel{QR}{=} [U\Sigma\ B]$.
2. Compute the SVD of R: $R \stackrel{SVD}{=} \tilde{U}\tilde{\Sigma}\tilde{V}^\top$.
3. Finally $U' = [U\ \ \tilde{B}]\tilde{U}$ and $\Sigma' = \tilde{\Sigma}$. If the desired number of basis vectors in U' is less than the number of non-zero singular values, then these excess vectors and singular values may be discarded.

The algorithm can also be made slightly faster, although somewhat more complicated, by modifying the arrangement of calculations in Step 1. Instead of computing the QR decomposition of $[U\Sigma\ \ B]$, $\tilde{B}$ and R can be obtained directly as follows: $\tilde{B} = \text{orth}(B - UU^\top B)$ and $R = \begin{bmatrix} \Sigma & U^\top B \\ 0 & \tilde{B}(B - UU^\top B) \end{bmatrix}$, where orth() performs orthogonalization, perhaps via QR. This reorganization, which follows from (2.1), avoids performing QR on the entire matrix $[U\Sigma\ \ B]$ (note that the columns corresponding to U are already orthogonal), instead only orthogonalizing $(B - UU^\top B)$, which is the component of B not already in the subspace U.

The computational advantage of the SKL algorithm over the naive approach is that it has space and time complexity that is constant in n, the number of training data seen so far. Specifically each update makes use of only the k largest singular values and basis vectors from the previous stage. This, together with the storage required for the m new images, reduces the space complexity to $O(d(k+m))$, down from $O(d(n+m)^2)$ with the naive approach. Similarly, the computational requirements are also reduced to $O(dm^2)$, versus $O(d(n+m)^2)$ for recomputing the entire SVD. More details and complexity analysis of the SKL algorithm are described in (Levy and Lindenbaum, 2000).

The problem with the SKL algorithm as stated above is that it makes no attempt to account for the sample mean of the training data, which changes over time as new data arrive. We will now show how this can be overcome. The essence of the approach is, at each update of the eigenbasis, to augment the new training data with an additional

vector carefully chosen to correct for the time-varying mean. We begin by proving the following lemma:

Lemma 1 *Let* $A = [\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_n]$, $B = [\mathbf{I}_{n+1}, \mathbf{I}_{n+2}, \ldots, \mathbf{I}_{n+m}]$ *be data matrices and* $C = [A\ B]$ *be their concatenation. Denote the means and scatter matrices of* A, B, C *as* $\bar{\mathbf{I}}_A$, $\bar{\mathbf{I}}_B$, $\bar{\mathbf{I}}_C$, *and* $\mathcal{S}_A$, $\mathcal{S}_B$, $\mathcal{S}_C$ *respectively. It can be shown that* $\mathcal{S}_C = \mathcal{S}_A + \mathcal{S}_B + \frac{nm}{n+m}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)^\top$.

In this lemma, we define a scatter matrix to be the outer product of the centered data matrix, for example $\mathcal{S}_B = \sum_{i=n+1}^{m}(I_i - \bar{I}_B)(I_i - \bar{I}_B)^\top$. *Thus a scatter matrix differs from the sample covariance matrix by only a scalar multiple* $\mathcal{S}_B = m\ \mathrm{cov}(B)$.

Proof of Lemma 1: By definition, $\bar{\mathbf{I}}_C = \frac{n}{n+m}\bar{\mathbf{I}}_A + \frac{m}{n+m}\bar{\mathbf{I}}_B$,
$\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C = \frac{m}{n+m}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)$, $\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C = \frac{n}{n+m}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)$ and,

$$\begin{aligned}
\mathcal{S}_C &= \textstyle\sum_{i=1}^{n}(\mathbf{I}_i - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_C)^\top + \sum_{i=n+1}^{n+m}(\mathbf{I}_i - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_C)^\top \\
&= \textstyle\sum_{i=1}^{n}(\mathbf{I}_i - \bar{\mathbf{I}}_A + \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_A + \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)^\top + \\
&\quad \textstyle\sum_{i=m+1}^{n+m}(\mathbf{I}_i - \bar{\mathbf{I}}_B + \bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_B + \bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)^\top \\
&= \mathcal{S}_A + n(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)^\top + \mathcal{S}_B + m(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)^\top \\
&= \mathcal{S}_A + \tfrac{nm^2}{(n+m)^2}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top + \mathcal{S}_B + \tfrac{n^2m}{(n+m)^2}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top \\
&= \mathcal{S}_A + \mathcal{S}_B + \tfrac{nm}{n+m}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top \qquad \square
\end{aligned}$$

From Lemma 1, and noting that

$$\begin{aligned}
&\mathcal{S}_B + \frac{nm}{n+m}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top \\
&\quad = \left[(B - \bar{\mathbf{I}}_B) \quad \sqrt{\frac{nm}{n+m}}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)\right]\left[(B - \bar{\mathbf{I}}_B) \quad \sqrt{\frac{nm}{n+m}}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)\right]^\top,
\end{aligned}$$

we can see that the SVD of $(C - \bar{I}_C)$ is equal to the SVD of the horizontal concatenation of $(A - \bar{I}_A)$, $(B - \bar{I}_B)$, and one additional vector $\sqrt{\frac{nm}{n+m}}(\bar{I}_B - \bar{I}_A)$. (The slight abuse of notation $(A - \bar{I}_A)$ is meant as a shorthand for the matrix $[(I_1 - \bar{I}_A) \ldots (I_n - \bar{I}_A)]$.) This motivates our new algorithm, appearing in Figure 2.1.

Let $A = [\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_n]$, $B = [\mathbf{I}_{n+1}, \mathbf{I}_{n+2}, \ldots, \mathbf{I}_{n+m}]$ be data matrices and $C = [A\ B]$ be their concatenation. Denote the mean of A as $\bar{\mathbf{I}}_A = \sum_{i=1}^{n} \mathbf{I}_i$.

Given U and Σ from the SVD of $(A - \bar{I}_A)$, as well as $\bar{I}_A$, n, and B, compute $\bar{I}_C$ as well as U' and Σ' from the SVD of $(C - \bar{I}_C)$:

1. Compute the mean vectors $\bar{I}_B = \frac{1}{m} \sum_{i=n+1}^{n+m} I_i$, and $\bar{I}_C = \frac{n}{n+m} \bar{I}_A + \frac{m}{n+m} \bar{I}_B$.
2. Form the matrix $\hat{B} = \left[(I_{m+1} - \bar{I}_B) \ \ldots \ (I_{n+m} - \bar{I}_B) \ \sqrt{\frac{nm}{n+m}} (\bar{I}_B - \bar{I}_A) \right]$.
3. Compute $\tilde{B} = \text{orth}(\hat{B} - UU^\top \hat{B})$ and $R = \begin{bmatrix} \Sigma & U^\top \hat{B} \\ 0 & \tilde{B}(\hat{B} - UU^\top \hat{B}) \end{bmatrix}$.
   Note that $\tilde{B}$ will be one column larger than in the SKL algorithm.
4. Compute the SVD of R: $R \overset{SVD}{=} \tilde{U} \tilde{\Sigma} \tilde{V}^\top$.
5. Finally $U' = [U \ \ \tilde{B}] \tilde{U}$ and $\Sigma' = \tilde{\Sigma}$.

Figure 2.1: The incremental PCA algorithm with mean update.

As can be seen, this algorithm shares the favorable complexity of the SKL algorithm, incurring only a small constant overhead to store, update, and correct for the changing sample mean.

Forgetting Factor

In numerous vision applications it is desirable to focus more on recently acquired images and less on earlier observations. For example, when tracking a target with a changing appearance, it is likely that recent observations will be more indicative of its appearance than would more distant ones. Down-weighting the contribution of earlier observations also plays an important role in online learning. As time progresses the observation history can become very large, to the point of overwhelming the relative contribution of each block of new data, rendering the learner 'blind' to changes in the observation stream.

One way to moderate the balance between old and new observations is to incorporate

a *forgetting factor* in the incremental eigenbasis update, as suggested by (Levy and Lindenbaum, 2000). To do this, at each update the previous singular values are multiplied by a scalar factor $f \in [0, 1]$, where $f = 1$ indicates no forgetting is to occur. Thus at Step 3 in Figure 2.1, $R = \begin{bmatrix} f\Sigma & U^\top \hat{B} \\ 0 & \tilde{B}(\hat{B} - UU^\top \hat{B}) \end{bmatrix}$, which is equivalent to taking the QR decomposition of $[fU\Sigma \;\; \hat{B}]$ instead of $[U\Sigma \;\; \hat{B}]$.

Although they propose the use of a forgetting factor, Levy and Lindenbaum do not provide any analysis as to its effect on the resulting eigenbasis. We address this with the following lemma:

Lemma 2 *A forgetting factor of f reduces the contribution of each block of data to the overall covariance modelled by an additional factor of f^2 at each SVD update.*

Proof of Lemma 2: When a forgetting factor of f is used, the incremental PCA algorithm in Figure 2.1 computes the left singular vectors U' and singular values Σ' of the matrix $[fU\Sigma \;\; \hat{B}]$. This is equivalent to computing the eigenvectors and (the square roots of) the eigenvalues of $[fU\Sigma \;\; \hat{B}][fU\Sigma \;\; \hat{B}]^\top$. Now

$$\begin{aligned} [fU\Sigma \;\; \hat{B}][fU\Sigma \;\; \hat{B}]^\top &= f^2 U\Sigma^2 U^\top + \hat{B}\hat{B}^\top \\ &= f^2 U\Sigma V^\top V \Sigma^\top U^\top + \hat{B}\hat{B}^\top \\ &= f^2 (A - \bar{I}_A)(A - \bar{I}_A) + \hat{B}\hat{B}^\top \\ &= f^2 \mathcal{S}_A + \mathcal{S}_B + ct, \end{aligned}$$

where ct is a correction term that adjusts the mean of the eigenbasis, and $\mathcal{S}_A$ and $\mathcal{S}_B$ are scatter matrices–a scalar times the covariance matrix–as defined in Lemma 1. □

Hence, after the k^{th} update of the eigenbasis, the block of m observations added during the j^{th} update ($j < k$) will have its covariance down-weighted by a factor of $f^{2(k-j)}$. The objective of PCA is to locate a subspace of dimension k that retains as much of the data covariance as possible (*i.e.* , maximizes the determinant of the projected data covariance $|U^\top \text{cov}(\text{Data})U|$ (Jolliffe, 2002)). Therefore it is a 'win' for PCA to select as basis vectors

directions of large covariance in recent data, at the expense of directions favored only by earlier data.

An important consideration not previously addressed is the effect of the forgetting factor on the mean of the eigenbasis. Since the contribution of the previously observed data to the covariance is decreased, it is necessary to also reduce its contribution to the resulting mean. When a forgetting factor of f is used, we propose the following modification to the mean update (Step 1 in Figure 2.1):

$$\bar{I}_C = \frac{fn}{fn+m}\bar{I}_A + \frac{m}{fn+m}\bar{I}_B$$

and at each update to compute the *effective* size of the observation history as $n \leftarrow fn+m$.

A benefit of incorporating the forgetting factor into the mean update is that the mean can still change in response to new observations, even as the actual number of observations approaches infinity. Specifically, using $n \leftarrow fn + m$, the effective number of observations will reach equilibrium at $n = fn + m$, or $n = m/(1 - f)$. For instance, when $f = 0.95$ and $m = 5$ new observations are included at each update, the effective size of the observation history will approach $n = 100$.

2.3.2 Sequential Inference Model

The visual tracking problem is cast as an inference task in a Markov model with hidden state variables. The state variable $\mathbf{X}_t$ describes the affine motion parameters (and thereby the location) of the target at time t. Given a set of observed images $\mathcal{I}_t = \{\mathbf{I}_1, \ldots, \mathbf{I}_t\}$, we aim to estimate the value of the hidden state variable $\mathbf{X}_t$. Using Bayes' theorem, we have the familiar result

$$p(\mathbf{X}_t|\,\mathcal{I}_t) \;\propto\; p(\mathbf{I}_t|\mathbf{X}_t) \int p(\mathbf{X}_t|\mathbf{X}_{t-1})\, p(\mathbf{X}_{t-1}|\,\mathcal{I}_{t-1})\, d\mathbf{X}_{t-1}. \tag{2.2}$$

The tracking process is governed by the observation model $p(\mathbf{I}_t|\mathbf{X}_t)$, where we estimate the likelihood of $\mathbf{X}_t$ observing $\mathbf{I}_t$, and the dynamical model between two states $p(\mathbf{X}_t|\mathbf{X}_{t-1})$.

The Condensation algorithm (Isard and Blake, 1996), based on factored sampling, approximates an arbitrary distribution of observations with a stochastically generated set of weighted samples. We use a variant of the Condensation algorithm to model the distribution over the object's location, as it evolves over time.

Dynamical Model

The location of a target object in an image frame can be represented by an affine image warp. This warp transforms the image coordinate system, centering the target within a canonical box such as the unit square, as illustrated in Figure 2.2. In this work the state at time t consists of the six parameters of an affine transformation $\mathbf{X}_t = (x_t, y_t, \theta_t, s_t, \alpha_t, \phi_t)$ where x_t, y_t, θ_t, s_t, α_t, ϕ_t, denote x, y translation, rotation angle, scale, aspect ratio, and skew direction at time t.

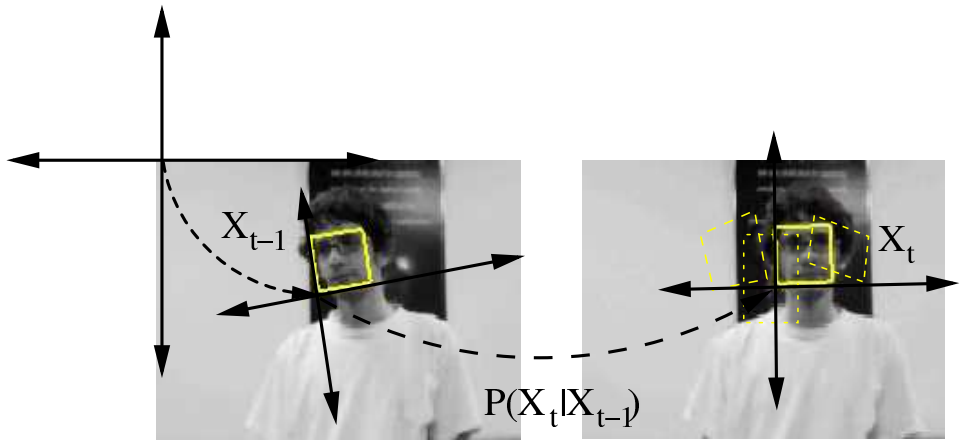

Figure 2.2: The model of dynamics. A location is represented by an affine transformation ($\mathbf{X}_{t-1}$), which warps the coordinate system so that the target lies within the unit square. Particles representing possible target locations $\mathbf{X}_t$ at time t are sampled according to $P(\mathbf{X}_t|\mathbf{X}_{t-1})$, which in this case is a diagonal-covariance Gaussian centered at $\mathbf{X}_{t-1}$.

To develop a tracker for generic applications, the dynamics between states in this space is modelled by Brownian motion. Each parameter in X_t is modelled independently

by a Gaussian distribution around its counterpart in X_{t-1}, and thus the motion between frames is itself an affine transformation. Specifically,

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t; \mathbf{X}_{t-1}, \mathbf{\Psi}) \tag{2.3}$$

where $\mathbf{\Psi}$ is a diagonal covariance matrix whose elements are the corresponding variances of the affine parameters, *i.e.* , σ_x^2, σ_y^2, σ_θ^2, σ_s^2, σ_α^2, σ_ϕ^2. These fixed parameters describe the kind of motion expected by the tracker. More complex dynamics can be modelled, such as first or second order dynamic systems, as well as other adaptive techniques for specific applications (North and Blake, 1998). Like all the other applications using particle filters, there is a trade off between the number of particles needed to be drawn (efficiency) and how well particle filters approximate the posterior distribution (effectiveness). With larger values in the diagonal covariance matrix $\mathbf{\Psi}$ and more particles, it is possible to track the object with higher precision at the cost of increased computation. In this project, we find a balance between these factors for efficient and effective visual tracking.

Observation Model

Since our goal is to use a representation to describe the "thing" that we are tracking, we model image observations using a probabilistic interpretation of principal component analysis (Roweis, 1997; Tipping and Bishop, 1999). Given an image patch $\mathbf{I}_t$ predicated by $\mathbf{X}_t$, we assume $\mathbf{I}_t$ was generated from a subspace of the target object spanned by U and centered at $\boldsymbol{\mu}$. The probability of a sample generated from a subspace, $p(\mathbf{I}_t|\mathbf{X}_t)$, is governed by a Gaussian distribution:

$$p(\mathbf{I}_t \,|\, \mathbf{X}_t) \;=\; \mathcal{N}(\mathbf{I}_t \,;\, \boldsymbol{\mu},\, UU^\top + \varepsilon I)$$

where I is an identity matrix, $\boldsymbol{\mu}$ is the mean, and εI term corresponds to the additive Gaussian noise in the observation process. It can be shown (Roweis, 1997) that in the limit as $\varepsilon \to 0$, this probability is proportional to the negative exponential

of the reprojection error or distance between $\mathbf{I}_t$ and its image on the subspace, *i.e.* $\exp(-||(\mathbf{I}_t - \boldsymbol{\mu}) - UU^\top(\mathbf{I}_t - \boldsymbol{\mu})||^2) \propto \mathcal{N}(\mathbf{I}_t; \boldsymbol{\mu}, UU^\top + \varepsilon I)$ as $\varepsilon \to 0$.

As written above, evaluating the probability requires inverting a $d \times d$ matrix, $(UU^\top + \varepsilon I)^{-1}$, which can be costly. Applying the Sherman-Morrison-Woodbury formula (Petersen and Pedersen, 2008) allows the expression to be rewritten as $(UU^\top + \varepsilon I)^{-1} = \varepsilon^{-1}(I - (1+\varepsilon)^{-1}UU^\top)$, which no longer requires matrix inversion. The resulting observation likelihood of $\mathbf{I}_t$, used to weight its corresponding particle in Condensation, is

$$p(\mathbf{I}_t \,|\, \mathbf{X}_t) \propto \exp\left(-\frac{1}{2\varepsilon}(\mathbf{I}_t - \boldsymbol{\mu})^\top (I - (1+\varepsilon)^{-1}UU^\top)(\mathbf{I}_t - \boldsymbol{\mu})\right). \tag{2.4}$$

2.3.3 Summary of the tracking algorithm

We now provide a summary of the proposed tracking algorithm in Figure 2.3. At the very beginning when the eigenbasis is empty (*i.e.* before the first update), our tracker works as a template based tracker. There is a natural trade-off between update frequency and speed of movement. Likewise, there is a trade-off between the number of particles and granularity of movement. We will discuss these implementation issues in the next section.

2.4 Implementation and Experiments

To evaluate empirical performance of the proposed tracker, we collected a number of videos recorded in indoor and outdoor environments where the targets change pose in different lighting conditions. Each video consists of 320×240-pixel grayscale images recorded at 30 frames per second, unless specified otherwise. Note that there exists large and unpredictable camera motion in the videos. For the eigenbasis representation, each target image region is resized to a 32×32 patch, and the number of eigenvectors used in all experiments is set to 16, though fewer eigenvectors can also work well. The forgetting term is empirically set to be 0.95, and the batch size for the eigenbasis update is set to 5

1. Locate the target object in the first frame, either manually or by using an automated detector, and use a single particle to indicate this location.

2. Initialize the eigenbasis U to be empty, and the mean μ to be the appearance of the target in the first frame. The effective number of observations so far is $n = 1$.

3. Advance to the next frame. Draw particles from the particle filter, according to the dynamical model.

4. For each particle, extract the corresponding window from the current frame, and calculate its weight, which is its likelihood under the observation model given by (2.4).

5. Store the image window corresponding to the most likely particle. When the desired number of new images have been accumulated, perform an incremental update (with a forgetting factor) of the eigenbasis, mean, and effective number of observations. In our experiments, the update is performed every fifth frame.

6. Go to step 3.

Figure 2.3: A summary of the proposed tracking algorithm.

as a trade-off between computational efficiency and effectiveness of modelling appearance change during fast motion. Implemented in MATLAB with MEX, our algorithm runs at 7.5 frames per second with 600 particles on a standard 2.8 GHz computer. Here we present selected tracking results, with more tracking results as well as videos available at http://www.cs.toronto.edu/~dross/phd/. Note that the results can be better viewed on high resolution displays or color printouts. Sample code and data sets are also available at the aforementioned website.

We begin by showing the results of our tracker on several sequences, then compare it qualitatively to two other state-of-the-art trackers. Next we evaluate and compare the trackers' quantitative performance, and empirically demonstrate the accuracy of our incremental PCA algorithm. We conclude with a discussion of the experimental results.

2.4.1 Experimental Results

We first tested our algorithm using a challenging video studied in (Jepson et al., 2003). The image sequence was downsampled by one-half, retaining only every other frame. Figure 2.4 shows the empirical results using our proposed method, where the first row of each panel shows the tracked objects (enclosed with rectangles) and the second row shows (from left to right) the subspace center, tracked image patch, residue, and reconstructed image using current eigenbasis. The red window shows the maximum a posteriori estimate of the particle filter, and green windows show the other particles whose weights are above a threshold. The eigenbasis images of the current subspace are shown in the third row of each panel (sorted according to their eigenvalues). Note that our method is able to track the target undergoing pose (#46, #185, #344, #376, #481), expression (#277, #398), and lighting (#344, #440) variation. Further, our method is able to track the target with temporary occlusion (#104) and structured appearance change such as glasses (#6, #185). Compared with the results reported in (Jepson et al., 2003), our method is able to efficiently learn a compact representation while tracking the target object without using

wavelets. All the eigenbases are constructed automatically from scratch and constantly updated to model the appearance of the target object, while it undergoes intrinsic and extrinsic changes. The eigenbases capture the appearance details of the target in different pose, expression, and with or without glasses,

Figure 2.4: Our method is able to track a human face undergoing pose, expression, appearance, and lighting change, as well as partial occlusion. The red window shows the maximum a posteriori estimate of the particle filter, and the green windows show the other particles with significant weight. Below the current video frame are four images that depict (from left to right) the mean of the eigenbasis, the current window selected by the tracker, the reprojection error, and the reconstruction of this window using the current eigenbasis. Finally, at the bottom are shown the eigenbasis images of the current subspace (sorted in decreasing order according to their eigenvalues). For the full result please refer to the accompanying video.

The second image sequence, shown in Figure 2.5, contains an animal doll moving in different pose, scale, and lighting conditions. Once initialized in the first frame, our algorithm is able to track the target object as it experiences large pose change (#65,

#272, #364, #521, #550, #609), a cluttered background (#65, #450, #521, #609), scale change (#65, #225), and lighting variation (#225, #364, #450, #609). Notice that the non-convex target object is localized within a rectangular window, and thus it inevitably contains some background pixels in its appearance representation. The results also show that our algorithm faithfully models the appearance of an arbitrary object, as shown in eigenbases and reconstructed images, in the presence of noisy background pixels. Nevertheless, our tracker eventually fails after frame 614 as a result of a combination of drastic pose and illumination change. Since the proposed algorithm is not limited to rectangular patches when specifying a region of interest for tracking, better results can be expected with more compact enclosing windows for specific targets.

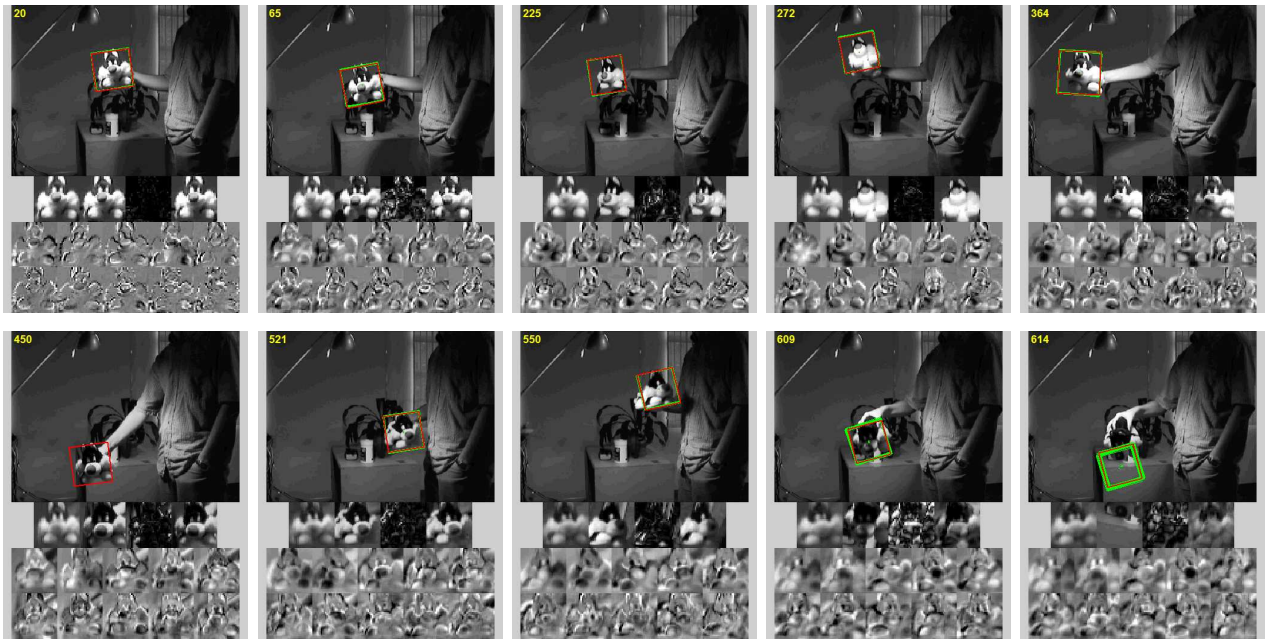

Figure 2.5: Our method is also able to track this animal doll for 614 frames, depite the effects of significant appearance variation resulting from a strong directional light source. (For a key to all of the various sub-figures, please refer to the caption of Figure 2.4.)

Figure 2.6 shows the tracking results using a challenging sequence, recorded at 15 frames per second with a moving digital camera, in which a person moves from a dark

room toward a bright area while changing his pose, moving underneath spotlights, changing facial expressions and taking off his glasses. Notice that there is also a large scale variation in the target relative to the camera. Even with the significant camera motion and low frame rate (which makes the motions between frames more significant, as when tracking fast-moving objects), our algorithm is able to track the target throughout the sequence, experiencing only temporary drifts. In contrast, most gradient or contour based trackers are not expected to perform well due to the large lighting variation, cast shadows, and unknown camera motion. With the use of a particle filter, our tracker is able to recover from temporary drifts due to a sudden and large pose change (between frames #166 and #167 in the accompanying video). Furthermore, the eigenbasis is constructed from scratch and is updated to reflect the appearance variation of the target object.

Figure 2.6: A person moves from a dark to a bright area, undergoing large lighting and pose changes. The images in the second row show the current sample mean, tracked region, reconstructed image, and the reconstruction error respectively. The third and fourth rows show the top 10 principal eigenvectors.

We recorded a sequence to evaluate our tracker in outdoor environments, where light-

ing conditions often change drastically. In it, a person walks underneath a trellis covered by vines, resulting in a significant variation in appearance due to cast shadows. As shown in Figure 2.7, the cast shadows change the appearance of the target face significantly (#96, #155, #170, #180, #223, #278). Furthermore, the pose and lighting variation combined with a low frame rate (15 fps) makes the tracking task rather challenging (#198, #303). Nevertheless, our algorithm is able track the target fairly accurately and robustly. In this sequence the forgetting term plays an important role, down-weighting the previously seen face images and focusing on the most recent ones, as the object appearance changes drastically.

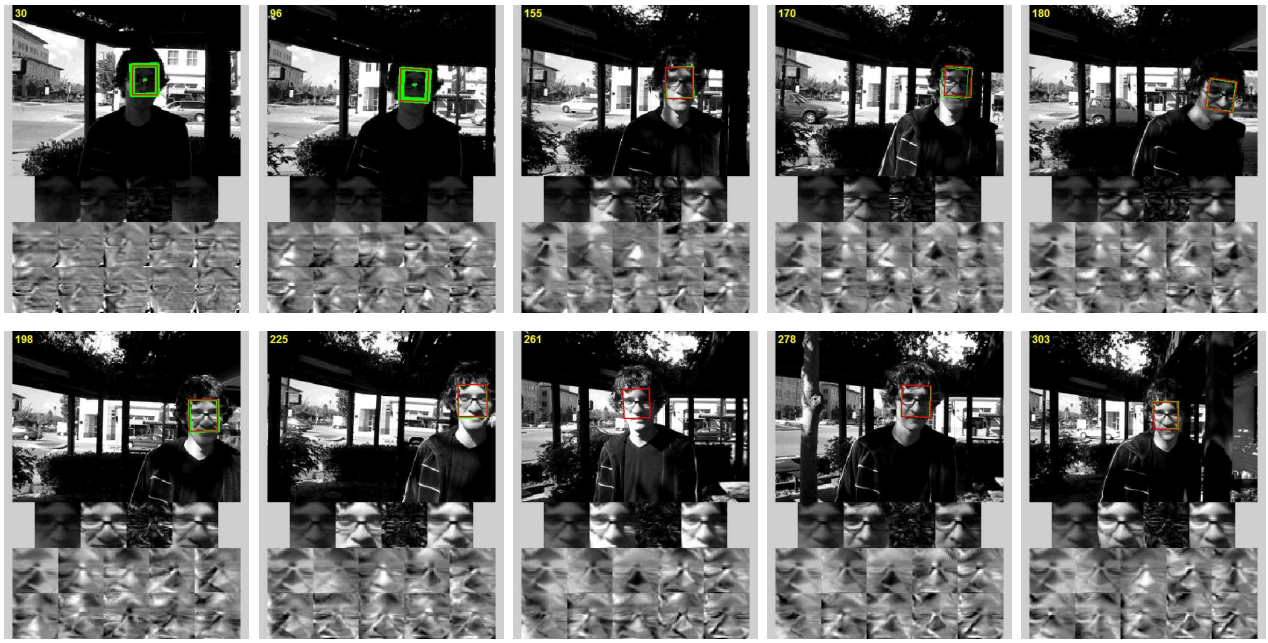

Figure 2.7: A person moves underneath a trellis with large illumination change and cast shadows while changing his pose. More results can be found in the project web page.

Figure 2.8 shows the results of tracking a moving vehicle, as it passes beneath a bridge and under trees. Although there is a sudden illumination change (#12, #184, #192, #232) in the scene, our tracker is able to track the target well. Our algorithm is also able to track objects in low resolution images, such as the sequence of a vehicle driving at night, shown in Figure 2.9. Despite the small size of the target relative to the

image, and the difficult illumination conditions, our algorithm is able to track the vehicle well.

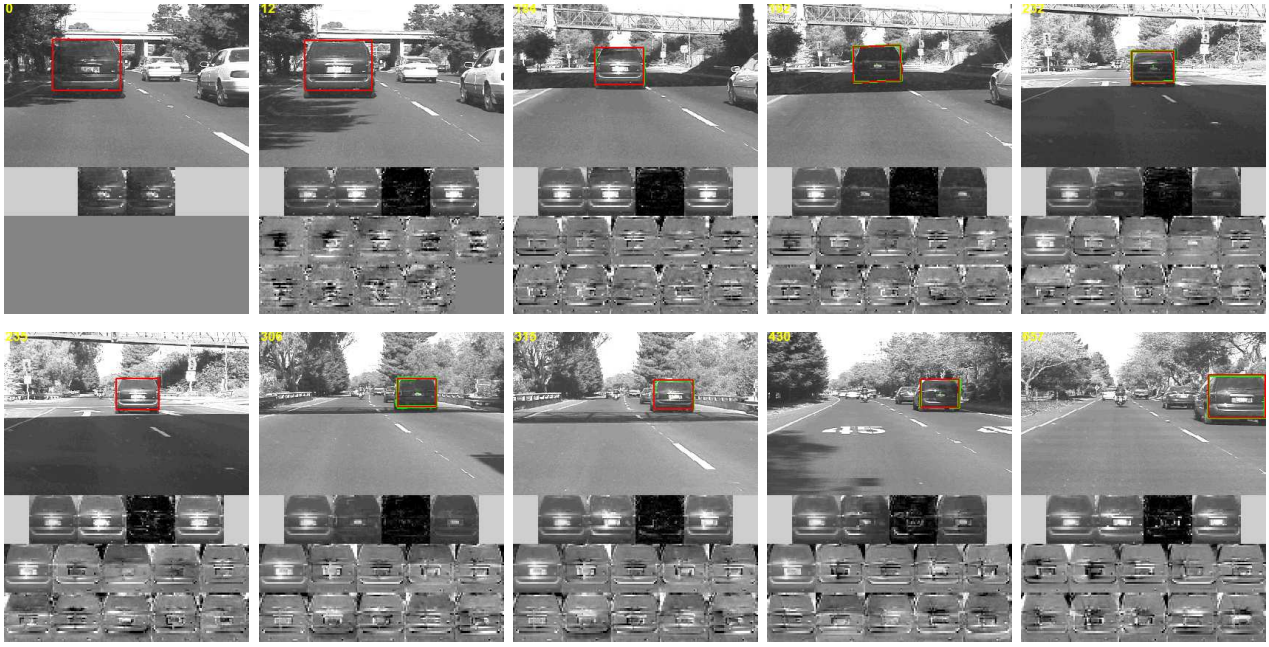

Figure 2.8: A vehicle moving underneath an overpass and trees. Our algorithm is able to track the target despite the large illumination variation.

2.4.2 Qualitative Comparison

As a qualitative benchmark, we ran two state-of-the-art algorithms, the WSL (Jepson et al., 2003) and Mean Shift (Comaniciu et al., 2003) trackers, on four of the sequences. The results are depicted in Figure 2.10. As can be seen in the figure (and corresponding videos), our method provides comparable performance to the WSL tracker. In the case of the first "Dudek" sequence, it is able to do so despite using a tighter target window around the face. However in the "animal doll" and "trellis", sequences, the WSL tracker proves to be more robust, continuing to track after our method fails. In both cases the targets experience drastic non-uniform changes in illumination from directed light sources. The WSL tracker gains a distinct advantage in these cases, based on its use of wavelet phase

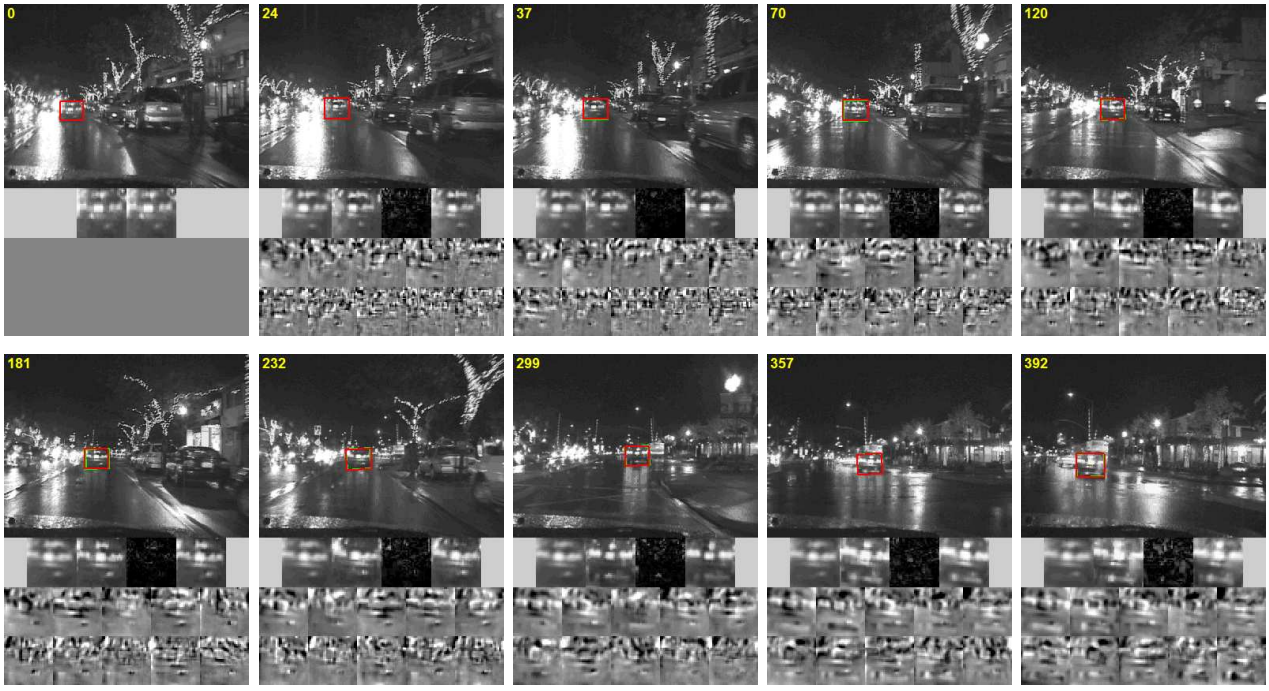

Figure 2.9: A vehicle moving in the night time with large illumination changes. Our algorithm is able to track the target when the images have low resolution and contrast.

features as an input representation. (It is possible that the performance of our tracker could also be improved by using a similar representation.) Further, our method not only tracks a target object, it also learns a compact low-dimensional representation which can be used for other applications such as recognition.

On the other hand, the Mean Shift tracker performs poorly, experiencing significant drift off the target objects. This can be attributed to the appearance model of the Mean Shift tracker, based on histograms of pixel intensities, which does not adapt over time, and is not sufficiently discriminative on these grayscale-only sequences. We expect that variants of the Mean Shift tracker using more sophisticated representations or adaptive multi-component models (Georgescu et al., 2004), would show improved performance.

Figure 2.10: A comparison of our tracker (indicated with a yellow box) with the WSL (Jepson et al., 2003) (shown in highlighted ellipse) and the Mean Shift (Comaniciu et al., 2003) (depicted by a green dashed box) on four video sequences.

2.4.3 Quantitative Analysis

To evaluate the tracking precision quantitatively, we tested the ability of our algorithm to consistently track seven facial feature points in the "Dudek" sequence. We compare our results with the manually labelled "ground truth" locations of the features, as initially presented in (Jepson et al., 2003).

To obtain estimates of the feature locations, we began by tracking the face, obtaining a sequence of similarity transformations approximately describing its motion from one frame to the next. For this image sequence, we used slightly larger variances, more particles (4000), and a forgetting factor of 0.99. Given the locations of the facial features in the first frame, we applied the sequence of transformations to these points, obtaining at each frame an estimate of where the features lay. Representative tracking results are shown in Figure 2.11, with red x's used to indicate our estimates of the feature locations, and yellow x's for the ground-truth positions.

Finally we computed the root mean square (RMS) error between the estimated locations of the features and the ground-truth. The error is plotted for each frame in Figure 2.12. For most frames our tracking results match the ground truth well, the largest errors occurring during brief occlusions or fast pose changes. The average RMS error of our method is 5.07 pixels per feature per frame, which is slightly better than the error of 5.2 pixels reported for the WSL tracker in (Jepson et al., 2003). In contrast, the Mean Shift tracker described in Section 2.4.2 has an average error of 48.7 pixels. Note that the errors in most frames are rather small and the errors in a few frames contribute most to the average RMS error.

Comparing the ability to track labeled features also allows us to quantitatively assess the contribution of the correct mean update and forgetting factor in our incremental algorithm to overall tracking performance. First, we re-ran the tracker without incrementally adapting the mean of the eigenbasis. The resulting average error increased to 5.86 pixels per feature per frame. Next, we removed the forgetting factor from the algo-

rithm (while using the correct mean update) and re-ran the tracker. This caused an even larger increase in error, to 7.70 pixels. Substituting our incremental algorithm with that of Hall et al. (2002), which lacks the forgetting factor, also produced an error of 7.70. These results demonstrate that the mean update and, particularly, the forgetting factor provide a measurable boost to tracking performance.

Figure 2.11: A person undergoing large pose, expression, appearance, and lighting change, as well as partial occlusions. The yellow crosses denote the ground truth data and the red crosses represent our tracking results.

To demonstrate the effectiveness of the proposed eigenbasis update algorithm in modelling object appearance, we compare the reconstruction results of our method to the conventional PCA algorithm, and to the incremental algorithm of Hall et al. (2002). For a fair comparison we do not use the forgetting factor in this experiment, so that the reconstruction error of each input image is treated equally by all algorithms. Unlike con-

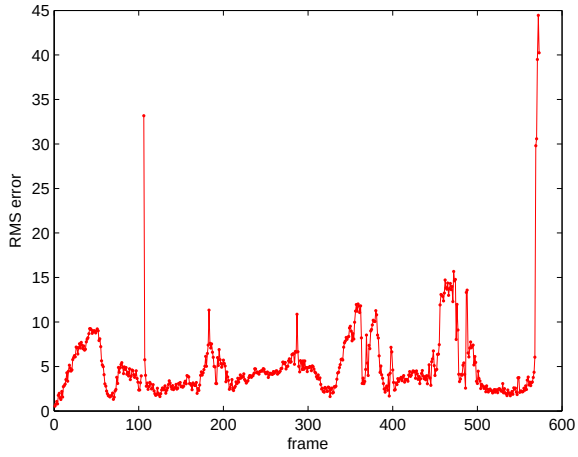

Figure 2.12: The RMS error at tracking feature points, for each frame in the "Dudek" sequence. The abrupt increases in error occur when there is temporary occlusion or motion blur.

ventional PCA, which constructs a subspace using all the frames in the video (*i.e.* batch processing), the incremental algorithms—Hall's and our own—update the subspace periodically as frames arrive. For this experiment, as with the tracker, we update the basis every five frames. At any given time the incremental algorithms retain only the top few eigenvectors, thereby providing an efficient method with a compact representation.

We used the "animal doll" sequence for experiments, extracting images of the target object from the first 605 frames of the sequence to use as training data. A selection of these images are depicted in the first row of Figure 2.13. The conventional batch PCA algorithm, our algorithm, and that of Hall et al. were used to construct bases consisting of 16 top eigenvectors. For both incremental algorithms this entailed 121 incremental updates, retaining only the top 16 eigenvectors after each update. When the learned bases were used to reconstruct the training images, batch PCA incurred a RMS reconstruction error of 7.93×10^{-2} per pixel, whereas the error of our algorithm was only slightly higher, at 8.03×10^{-2}. The reconstructed images using the batch PCA algorithm and our algorithm are shown in rows 2 and 4 of Figure 2.13 respectively, and rows 3 and

5 contain the corresponding residue images.

In comparison, Hall's algorithm achieved the same reconstruction error as our own, 8.03×10^{-2}, however its runtime, averaged over 100 repetitions, was 38% greater than that of our algorithm. (The results of Hall's algorithm are not included in the figure since, when the forgetting factor is not used, they are visually indistinguishable from our own.) The batch PCA algorithm takes on average 6 times longer than our incremental algorithm, even after we rearrange the computation to calculate the eigenvectors of the Gram matrix ($X^\top X$) rather than the covariance matrix ($XX^\top$), as described in (Murase and Nayar, 1995).

Thus, from these experiments we can conclude that our incremental eigenbasis update method is able to effectively model the object appearance without losing detailed information, at a cost appreciably less than that of Hall et al.'s algorithm.

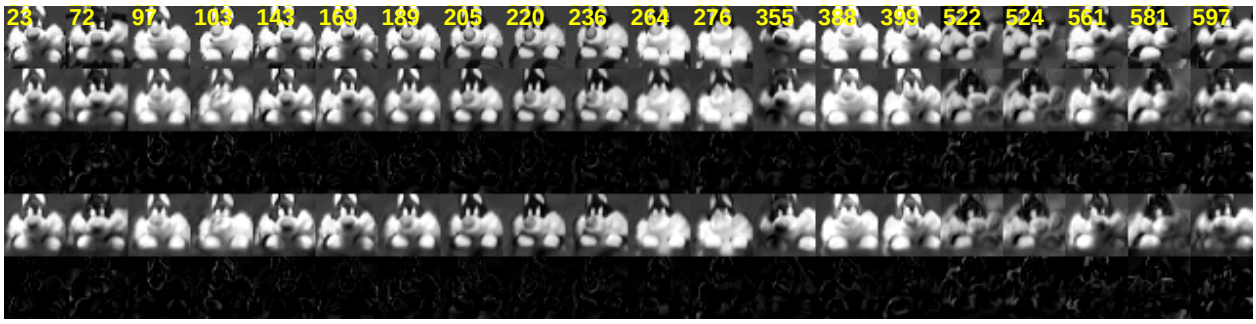

Figure 2.13: The first row shows a selection of test images. The second and fourth rows show the reconstructions of these images using the conventional batch algorithm and our incremental algorithm, respectively. Their corresponding residues are presented in the third and fifth rows.

2.4.4 Discussion

The robust tracking performance of our algorithm can be attributed to several factors. One reason is that our incremental eigenbasis learning approach exploits the local linear-

ity of appearance manifold for matching targets in consecutive frames. It is well known that the appearance of an object undergoing pose change can be modelled well with a view-based representation (Murase and Nayar, 1995). Meanwhile at fixed pose, the appearance of an object in different illumination conditions can be approximated well by a low dimensional subspace (Belhumeur and Kreigman, 1997). Our empirical results show that these variations can be learned online without any prior training, and that the changes caused by cast and attached shadows can still be approximated by a linear subspace to limited extent. Consequently the appearance of an object undergoing illumination and pose variation can be approximated by a local subspace within a short span of time, which in turns facilitates the tracking task. Notice that at any time instant, it suffices to use an eigenbasis to account for appearance variation if the object motion or illumination change is not instantly drastic. This work demonstrates that a tracker based on the idea of an incremental eigenbasis update can be both efficient and perform well empirically when the appearance change is gradual. A few additional failure cases for this algorithm can be seen at project the web site, mentioned earlier. Typically, failures happen when there is a combination of fast pose change and drastic illumination change.

In this project we do not directly address the partial occlusion problem. Empirical results show that temporary and partial occlusions can be handled by our method through constant update of the eigenbasis and the robust error norm. Nevertheless situations arise where we may have prior knowledge of the object being tracked, and can exploit such information for better occlusion handling.

2.5 Conclusions and Future Work

We have presented an appearance-based tracker that incrementally learns a low dimensional eigenbasis representation for robust object tracking while the target undergo pose, illumination and appearance changes. Whereas most algorithms operate on the premise

that the object appearance or ambient environment lighting conditions do not change as time progresses, our method adapts the model representation to reflect appearance variation of the target, thereby facilitating the tracking task. In contrast to the existing incremental subspace methods, our eigenbasis update method updates the mean and eigenbasis accurately and efficiently, and thereby learns to faithfully model the appearance of the target being tracked. Our experiments demonstrate the effectiveness of the proposed tracker in indoor and outdoor environments where the target objects undergo large pose and lighting changes.

Although our tracker performs well, it occasionally drifts from the target object. With the help of particle filters, the tracker often recovers from drifts in the next few frames when a new set of samples is drawn. For specific applications, better mechanisms to handle drifts could further enhance robustness of the proposed algorithm. The current dynamical model in our sampling method is based on a Gaussian distribution, but for certain specific applications the dynamics could be learned from exemplars for more efficient parameter estimation. Our algorithm can also be extended to construct a set of eigenbases for modelling nonlinear aspects of appearance variation more precisely and automatically. We aim to address these issues in our future work.

Chapter 3

Combining Discriminative Features to Infer Complex Trajectories

In this chapter we propose a new model for the probabilistic estimation of continuous state variables from a sequence of observations, such as tracking the position of an object in video. This mapping is modeled as a product of dynamics experts (features relating the state at adjacent time-steps) and observation experts (features relating the state to the image sequence). Individual features are flexible in that they can switch on or off at each time-step depending on their inferred relevance (or on additional side information), and discriminative in that they need not model the full generative likelihood of the data. When trained conditionally, this permits the inclusion of a broad range of rich features, such as features relying on observations from multiple time-steps. Furthermore, the relevance of these features can be learned from labelled sequences.

3.1 Introduction

Many real-world problems involve estimating a time series of continuous state vectors from a sequence of high-dimensional observations. Examples include inferring a trajectory of stock values based on the evolution of various economic indicators; tracking a

patient's vital health signs through a myriad of symptoms; and tracking the trajectory of a moving object in video. A standard probabilistic approach is to fit the observations with a generative state-space model (SSM). These models propose that the state is a latent variable which evolves over time, and at each step is responsible for generating a noisy observation. State estimates are obtained from observations by inverting the probability model via Bayes' rule. A canonical example of a SSM is the Kalman filter, which models both the state dynamics and observations as linear functions of the state, corrupted by Gaussian noise, which leads to a Gaussian posterior distribution over the state at any time. Extensions to the Kalman filter allow for non-linearities in the state dynamics and observation functions, and for multi-modal posterior state distributions.

Although highly successful, SSMs suffer some disadvantages. First, for computational tractability, SSMs usually assume that observations are conditionally independent of each other given the state. Thus the estimate of the state at a particular time can directly depend only on the observations at that same time (and the previous state), precluding the direct inclusion of evidence derived from observations across a range of time. Second, the relationship between state and observation at every time-step in an SSM is mediated through a single, identical likelihood function, which must generate the entire high-dimensional observation given only the value of the state. Crafting such a likelihood can be challenging, since it requires the ability to accurately model all aspects of the observation, including those that are irrelevant with respect to predicting the state.

An alternative approach, which we pursue here, is to directly model the conditional (posterior) distribution of the states given the observations. We propose fitting the posterior with a weighted log-linear combination of dynamics features (relating states at different time-steps) and observation features (relating the state to the observation sequence). Features are discriminative, leveraged to predict the state from the observations, thus avoiding the problem of explaining the high-dimensional observations faced by generative likelihoods. Using a conditional model also removes the need to assume inde-

pendence of observations, hence each feature may incorporate evidence from any number of observations. Additionally, a wide variety of observation features may be combined and the system can learn, through supervised training, which features are relevant for any given task. Our system also includes the ability at each time-step to switch between different dynamics features, and to selectively shut off unreliable observation features.

A canonical application of this approach involves tracking a moving object in a video as it follows a complicated trajectory. Consider watching a basketball game, and focusing simply on trying to follow the ball. Generating a full description of the scene, including the complex interaction of all the players, based on the ball position at any time is hopeless. However, obtaining a estimate of the ball's location from the image is considerably easier, as features such as colour and shape can be highly discriminative. In addition, understanding the basic dynamics of the ball motion can be useful in tracking it even as it disappears behind some players.

We begin in Section 3.2 with a detailed description of our model, followed by details on inference and learning (Section 3.3). We relate our model to similar approaches in Section 3.5. Finally, in Section 3.6, we apply our model to a realistic tracking problem—estimating the position of a basketball in a video—using a number of different dynamics and observation features.

3.2 Model

Given a sequence of observations $\mathbf{Y}$ and corresponding sequence of states $\mathbf{X}$, we construct a model of the conditional distribution of $\mathbf{X}$ given $\mathbf{Y}$. The model combines a set of dynamics features $f_j(\mathbf{x}_{t-1}, \mathbf{x}_t)$ for $j = 1, \ldots, J$, and observation features $g_k(\mathbf{x}_t, \mathbf{Y})$ for $k = 1, \ldots, K$. The basic model then combines these features to provide a description of the conditional distribution:

$$\mathrm{P}(\mathbf{X}|\mathbf{Y}) \propto \exp\left(\sum_{t=2}^{T}\sum_{j=1}^{J} f_j(\mathbf{x}_{t-1}, \mathbf{x}_t) + \sum_{t=1}^{T}\sum_{k=1}^{K} g_k(\mathbf{x}_t, \mathbf{Y})\right).$$

In our model, both the dynamics and observation features can be viewed as functions that predict the state $\mathbf{x}_t$. That is, associated with each dynamics feature $f_j(\mathbf{x}_{t-1}, \mathbf{x}_t)$ is a function $\phi_j(\mathbf{x}_{t-1})$, which predicts the state at time t given the state at time $t-1$. Each f_j then computes the distance between $\mathbf{x}_t$ and its predicted value $\phi_j(\mathbf{x}_{t-1})$, which is scaled by a learned parameter $\boldsymbol{\alpha}_j$. Similarly, each observation feature $g_k(\mathbf{x}_t, \mathbf{Y})$ has an associated function $\gamma_k(\mathbf{Y}, t)$, predicting $\mathbf{x}_t$ from observations, and a scaling parameter $\boldsymbol{\beta}_k$:

$$f_j(\mathbf{x}_{t-1}, \mathbf{x}_t) = -\frac{1}{2}\left(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1})\right)^\top \boldsymbol{\alpha}_j \left(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1})\right)$$
$$g_k(\mathbf{x}_t, \mathbf{Y}) = -\frac{1}{2}\left(\mathbf{x}_t - \gamma_k(\mathbf{Y}, t)\right)^\top \boldsymbol{\beta}_k \left(\mathbf{x}_t - \gamma_k(\mathbf{Y}, t)\right).$$

The range of possible functions $\phi_j()$ and $\gamma_k()$ is broad, including any off-the-shelf method of predicting the state from other states or observations. In this work we will restrict our attention to linear functions for the dynamics:

$$\phi_j(\mathbf{x}_{t-1}) = \mathbf{T}_j \mathbf{x}_{t-1} + \mathbf{d}_j.$$

For example, in describing the motion of object in two dimensions, $\mathbf{x}_t$ can include a pair of components for its position, velocity, and acceleration, and $(\mathbf{T}_j, \mathbf{d}_j)$ could correspond to constant-velocity and no acceleration, or constant acceleration.

When including a large number of features, it is likely that at any given time, some of them give very poor predictions, and their contributions should be disregarded. This problem can be addressed by making the set of features flexible: at each time-step features can be turned off (meaning that their prediction will not be included in the state estimate) based on their inferred relevance. This is accomplished through the introduction of hidden binary *switch* variables, u_{jt} and v_{kt}, one for each feature at each time-step.

Obtaining an accurate estimate of the state therefore is highly dependent on appropriately setting the switches, to only include the relevant features in the state representation. One piece of information potentially relevant to determining the switches involves evalu-

Table 3.1: A summary of notation

$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_T]$	state sequence
$\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_T]$	observation sequence
$f_j(\mathbf{x}_{t-1}, \mathbf{x}_t)$	j^{th} dynamics feature function,
$\boldsymbol{\alpha}_j$	its parameter
$\phi_j(\mathbf{x}_{t-1})$	its prediction function $\phi_j(\mathbf{x}_{t-1}) = \mathbf{T}_j\mathbf{x}_{t-1} + \mathbf{d}_j$
$\mathbf{T}_j$	its linear dynamics model (matrix)
$\mathbf{d}_j$	its linear dynamics model (translation vector)
u_{jt}	binary switch on dynamics
$\mathcal{F}_j(\mathbf{Y}, t)$	dynamics switch potential
$g_k(\mathbf{x}_t, \mathbf{Y})$	k^{th} observation feature function,
$\boldsymbol{\beta}_k$	its parameter
$\gamma_k(\mathbf{Y}, t)$	its prediction function
v_{kt}	binary switches on observations
$\mathcal{G}_k(\mathbf{Y}, t)$	observation switch potential

ating the agreement between the feature predictions; intuitively a feature making a very divergent prediction can be switched off. Often, there is additional information available in the observations to suggest which dynamics/observation features might be relevant. This side information is captured by including learned potential functions for each switch, $\mathcal{F}_j$ and $\mathcal{G}_k$, which again can be off-the-shelf classifiers, trained discriminatively in the same framework.

Putting these together, we arrive at the following log-probability of $\mathbf{X}$ given $\mathbf{Y}$:

$$\mathcal{L} = \log \sum_{\mathbf{u},\mathbf{v}} \exp\left(\sum_{t,j} f_j(\mathbf{x}_{t-1}, \mathbf{x}_t) u_{jt} + \sum_{t,k} g_k(\mathbf{x}_t, \mathbf{Y}) v_{kt} + \sum_{t,j} \mathcal{F}_j(\mathbf{Y}, t) u_{jt} + \sum_{t,k} \mathcal{G}_k(\mathbf{Y}, t) v_{kt} \right) - \log Z(\mathbf{Y}) \quad (3.1)$$

where $Z(\mathbf{Y})$ is the distribution's normalizing function, or *partition function*. The notation used in this equation is detailed in Table 3.1, and the corresponding factor graph illustrated in Figure 3.1. Essentially, the log probability of the state sequence $\mathbf{X}$ given the observation sequence $\mathbf{Y}$ is a linear combination of four probability potentials. The dynamics features, $f_j(\mathbf{x}_{t-1}, \mathbf{x}_t)$, each assess the probability that the state begins at $\mathbf{x}_{t-1}$ at time $t-1$ and moves to $\mathbf{x}_t$ at time t. Similarly, the observation features, $g_k(\mathbf{x}_t, \mathbf{Y})$, each assess the probability that the state at time t is $\mathbf{x}_t$, given the collection of observations $\mathbf{Y}$. Finally the switch potential $\mathcal{F}_j(\mathbf{Y}, t)$ assess the probability that dynamics feature f_j is giving an accurate estimate for the transition from time $t-1$ to time t, and $\mathcal{G}_k(\mathbf{Y}, t)$ that the observation feature g_k is giving a reliable estimate at time t.

Note that because the switches are hidden, when we integrate over our uncertainty for the switches we effectively get a mixture-of-Gaussian prediction for the posterior state distribution at time t, which allows us to elegantly capture multimodality. Further, during inference, the posterior distribution over switches will capture the *probability* that a given feature is going to be of use at that point in the sequence.

The non-switching or "non-flexible" version of this model—where u_{jt} and v_{kt} are fixed constant—is an interesting special case. Because the dynamics and observation

features are quadratic in $\mathbf{X}$, the resulting conditional distribution is Gaussian. This provides some advantages: exact inference, partition function, and gradient computations can be done efficiently, and learning (with respect to $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_j$) becomes a convex problem. Practically, however, there are two considerable disadvantages. First, because the Gaussian is unimodal, the resulting state distribution will be unimodal at all time-steps; this can lead to an inability to recover from errors in the state prediction. Second, the prediction is no longer robust, which means that only dynamic features which are all simultaneously applicable can be included, and observation features must be always accurate.

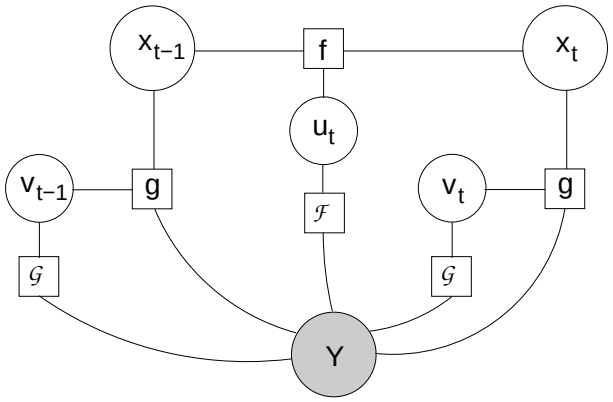

Figure 3.1: Factor graph of the model for two time-steps.

3.3 Inference

Given a sequence of observations, inferring the corresponding state sequence consists of computing the probability distribution $\mathrm{P}(\mathbf{X}|\mathbf{Y})$. Performing this calculation directly is infeasible, unfortunately, since it requires marginalization over all possible joint settings of the hidden variables, u_{jt} and v_{kt}, and there are hidden variables for each time-step. However, several variational and approximation schemes readily apply to this formulation. Here we focus on a particular MCMC method that exploits special structure in the

model to allow efficient approximate inference.

Given the hidden variables, the state sequence $\mathbf{X}$ forms an (undirected) linear-Gaussian Markov chain, thus $P(\mathbf{X}|\mathbf{U},\mathbf{V},\mathbf{Y})$ can be readily computed. Similarly, given the state sequence, the switches are conditionally independent, so inference of $P(\mathbf{U},\mathbf{V}|\mathbf{X},\mathbf{Y})$ is easy. From these facts, we arrive at a simple Gibbs sampling method for drawing samples from $P(\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y})$.

1. Obtain an initial estimate $\hat{\mathbf{U}},\hat{\mathbf{V}}$ of the switch variables. For example, these can be based on the side-information provided by features $\mathcal{F}_j$ and $\mathcal{G}_k$.

2. Infer $P(\mathbf{X}|\hat{\mathbf{U}},\hat{\mathbf{V}},\mathbf{Y})$, a Gaussian in $\mathbf{X}$, and draw from it a state sequence sample $\hat{\mathbf{X}}$.

3. Infer $P(\mathbf{U},\mathbf{V}|\hat{\mathbf{X}},\mathbf{Y})$, and from it draw samples of the switches $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$.

4. Goto 2, and repeat this sampling procedure for the desired number of iterations.

We now present a message-passing scheme for inferring the state given the switches, followed by a pair of simple equations for inferring the switches given the state. Each iteration of this scheme (and of learning) has a computational cost that scales linearly in the length of the sequence and the number of observation and dynamics features, but, like Kalman smoothing, scales cubically in the dimensionality of the state. A full derivation of this algorithm is presented in Appendix B.

3.3.1 Inferring State X given Switches U, V

Given the switches and observations, the belief propagation algorithm can be used to exactly compute the marginal $P(\mathbf{x}_t|\mathbf{U},\mathbf{V},\mathbf{Y})$ and pairwise marginal $P(\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{U},\mathbf{V},\mathbf{Y})$ distributions. These can be used to draw samples of the state sequence, as well as to compute the expectations (3.2) which will be required for learning. Note that because

the features are quadratic in $\mathbf{X}$, the marginals and pairwise marginals will be Gaussian distributions.

Inference using belief propagation requires a two-phase message passing schedule, much like the Kalman smoother; messages are passed forward, from the beginning of the state sequence to the end, and backward in the opposite direction. Each message consists of a Gaussian distribution, with mean vector μ and precision matrix $\boldsymbol{\tau}$ (which, for notational convenience, we will use in place of the inverse covariance matrix). As in Kalman smoothing, the messages can be written recursively, each in terms of the message preceding it. This algorithm differs from Kalman smoothing only in the additional bookkeeping required to accommodate varying switched dynamics between each time-step, and products of switched observation features. Message passing can be broken into four steps: forward prediction of $\mathbf{x}_t$ given $\mathbf{x}_{t-1}$, forward correction to incorporate observation features at time t, backward prediction of $\mathbf{x}_t$ given $\mathbf{x}_{t+1}$, and backward correction.

We first define a number of terms which will appear several times in the message-passing equations. Note that because they depend on the switches, these terms must be recomputed at each time-step.

$$\mathbf{A} = \sum_j \alpha_j u_{jt} \qquad \mathbf{B} = \sum_k \beta_k v_{kt}$$

$$\alpha T = \sum_j \alpha_j \mathbf{T}_j u_{jt} \qquad T\alpha T = \sum_j \mathbf{T}_j{}^\top \alpha_j \mathbf{T}_j u_{jt}$$

$$\alpha d = \sum_j \alpha_j \mathbf{d}_j u_{jt} \qquad T\alpha d = \sum_j \mathbf{T}_j \alpha_j \mathbf{d}_j u_{jt}$$

$$\mathbf{y}'_t = \sum_k \beta_k \gamma_k(\mathbf{Y}, t) v_{kt}$$

The first phase begins with an initial estimate of the state at time-step 1, such as $\mathcal{N}(\mathbf{x}_1|\ \sum_k g_k(\mathbf{x}_1, \mathbf{Y}) v_{k1},\ \sum_k \beta_k v_{k1})$. For each subsequent time-step, we compute a predicted estimate $\mathcal{N}(\mathbf{x}_t|\ \boldsymbol{\mu}_{t|t-1},\ \boldsymbol{\tau}_{t|t-1})$ of the state based on our estimate for the previous time-step, and a corrected esimate $\mathcal{N}(\mathbf{x}_t|\ \boldsymbol{\mu}_t^f,\ \boldsymbol{\tau}_t^f)$ which incorporates information from $g_k(\mathbf{x}_t, \mathbf{Y})$. These forward updates are:

Forward Prediction:

$$\boldsymbol{\tau}_{t|t-1} = \mathbf{A} - \alpha T(T\alpha T + \boldsymbol{\tau}^f_{t-1})^{-1}\alpha T^\top$$

$$\boldsymbol{\mu}_{t|t-1} = (\boldsymbol{\tau}_{t|t-1})^{-1}[\alpha T(T\alpha T + \boldsymbol{\tau}^f_{t-1})^{-1}(\boldsymbol{\tau}^f_{t-1}\boldsymbol{\mu}^f_{t-1} - T\alpha d) + \alpha d]$$

Incorporating Evidence (forward correction):

$$\boldsymbol{\tau}^f_t = \boldsymbol{\tau}_{t|t-1} + \mathbf{B}$$

$$\boldsymbol{\mu}^f_t = (\boldsymbol{\tau}^f_t)^{-1}(\boldsymbol{\tau}_{t|t-1}\boldsymbol{\mu}_{t|t-1} + \mathbf{y}'_t)$$

The second phase begins with our corrected estimate of the final state obtained from the forward pass $\mathcal{N}(\mathbf{x}_T|\, \boldsymbol{\mu}^f_T, \boldsymbol{\tau}^f_T)$ and works backwards, computing predicted $\mathcal{N}(\mathbf{x}_t|\, \boldsymbol{\mu}_{t|t+1},\, \boldsymbol{\tau}_{t|t+1})$ and corrected $\mathcal{N}(\mathbf{x}_t|\, \boldsymbol{\mu}^b_t,\, \boldsymbol{\tau}^b_t)$ estimates.

Backward Prediction:

$$\boldsymbol{\tau}_{t|t+1} = T\alpha T - \alpha T^\top(\mathbf{A} + \boldsymbol{\tau}^b_{t+1})^{-1}\alpha T$$

$$\boldsymbol{\mu}_{t|t+1} = (\boldsymbol{\tau}_{t|t+1})^{-1}[\alpha T^\top(\mathbf{A} + \boldsymbol{\tau}^b_{t+1})^{-1}(\boldsymbol{\tau}^b_{t+1}\boldsymbol{\mu}^b_{t+1} + \alpha d) - T\alpha d]$$

Backward Correction:

$$\boldsymbol{\tau}^b_t = \boldsymbol{\tau}_{t|t+1} + \mathbf{B}$$

$$\boldsymbol{\mu}^b_t = (\boldsymbol{\tau}^b_t)^{-1}(\boldsymbol{\tau}_{t|t+1}\boldsymbol{\mu}_{t|t+1} + \mathbf{y}'_t)$$

Finally, the marginal distribution of $\mathbf{x}_t$, $\mathcal{N}(\mathbf{x}_t|\, \boldsymbol{\mu}_t,\, \boldsymbol{\tau}_t)$ can be obtained by multiplying all messages coming into it:

$$\boldsymbol{\tau}_t = \boldsymbol{\tau}_{t|t-1} + \mathbf{B} + \boldsymbol{\tau}_{t|t+1}$$

$$\boldsymbol{\mu}_t = (\boldsymbol{\tau}_t)^{-1}(\boldsymbol{\tau}_{t|t-1}\boldsymbol{\mu}_{t|t-1} + \mathbf{y}'_t + \boldsymbol{\tau}_{t|t+1}\boldsymbol{\mu}_{t|t+1}).$$

The pairwise marginal distribution is obtained by multiplying together the forward message into $\mathbf{x}_{t-1}$, the backward message into $\mathbf{x}_t$, and an additional factor arising from the dynamics features. The resulting mean is simply the concatenation of the marginal means, $(\boldsymbol{\mu}_{t-1}, \boldsymbol{\mu}_t)$, and the precision is sum of the precisions from the messages and dynamics factor

$$\boldsymbol{\tau}_{t-1,t} = \begin{bmatrix} \boldsymbol{\tau}_{t-1|t-2} + \mathbf{B} + T\alpha T & -\alpha T^\top \\ -\alpha T & \mathbf{A} + \mathbf{B} + \boldsymbol{\tau}_{t|t+1} \end{bmatrix}.$$

3.3.2 Inferring Switches $\mathbf{U}, \mathbf{V}$ given State $\mathbf{X}$

As mentioned above, given the state sequence the switch variables are independent, thus $\mathrm{P}(\mathbf{U}, \mathbf{V}|\mathbf{X}, \mathbf{Y})$ factorizes into a product of simple distributions.

The posteriors of the observation switches v_{kt} are independent Bernoulli distributions, with probability

$$\mathrm{P}(v_{kt} = 1) = \sigma\left(g_k(\mathbf{x}_t, \mathbf{Y}) + \mathcal{G}_k(\mathbf{Y}, t)\right)$$

where $\sigma()$ is the *logistic* function $\sigma(x) = 1/(1 + e^{-x})$. The posterior distribution of the dynamics switches u_{jt} at each time-step is Multinomial—a discrete choice over J options. The probability that switch j is on at time t is

$$\mathrm{P}(u_{jt} = 1) = \frac{\exp(f_j(\mathbf{x}_{t-1}, \mathbf{x}_t) + \mathcal{F}_j(\mathbf{Y}, t))}{\sum_{j'} \exp(f_{j'}(\mathbf{x}_{t-1}, \mathbf{x}_t) + \mathcal{F}_{j'}(\mathbf{Y}, t))}.$$

3.4 Learning

The parameters of the feature functions, $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_k$, can be learned using an application of the Contrastive Divergence algorithm (Hinton, 2002). Contrastive Divergence is an approximate gradient descent in parameter space, used in undirected graphical models with intractable partition functions. Specifically, analytic differentiation of the log-likelihood, (3.1), with respect to $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_k$ results in the following expressions:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_j} =& E_{\mathrm{P}(\mathbf{U},\mathbf{V}|\mathbf{X},\mathbf{Y})}\left[\sum_t (\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))^\top u_{jt}\right] \\
& - E_{\mathrm{P}(\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y})}\left[\sum_t (\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))^\top u_{jt}\right], \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}_k} =& E_{\mathrm{P}(\mathbf{U},\mathbf{V}|\mathbf{X},\mathbf{Y})}\left[\sum_t (\mathbf{x}_t - \gamma_k(\mathbf{Y}, t))(\mathbf{x}_t - \gamma_k(\mathbf{Y}, t))^\top v_{kt}\right] \\
& - E_{\mathrm{P}(\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y})}\left[\sum_t (\mathbf{x}_t - \gamma_k(\mathbf{Y}, t))(\mathbf{x}_t - \gamma_k(\mathbf{Y}, t))^\top v_{kt}\right].
\end{aligned} \tag{3.2}$$

Thus, the gradient of the log-probability with respect to $\boldsymbol{\alpha}_j$ is the difference between two expectations. The first expectation, known as the *positive phase*, computes the

expectation of $\sum_t (\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))^\top u_{jt}$ given the observation sequence $\mathbf{Y}$ and corresponding ground truth state sequence $\mathbf{X}$. (This quantity can be thought of as the uncentered second moment (covariance) of the error made by predictor ϕ_j, in cases where it is switched on.) As was seen in Section 3.3.2, the switch variables are conditionally independent given $\mathbf{X}$ and $\mathbf{Y}$, so their expected values are readily obtained, and the resulting expectation is simply $\sum_t (\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1}))^\top \mathrm{E}[u_{jt}|\mathbf{X}, \mathbf{Y}]$.

The second expectation, known as the *negative phase*, computes the expected value of the same quantity, but this time given only the observations $\mathbf{Y}$. This is the same computation required during inference, as described in Section 3.3, and is likewise infeasible. To deal with this, Contrastive Divergence proposes an approximation very much like the one used for inference, but that makes use of the ground truth labels $\mathbf{X}$, which are available at training time. Specifically, samples of $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{X}$ given $\mathbf{Y}$ are drawn according the the following procedure, and are then used to obtain an approximate Monte Carlo estimate of the negative phase expectation.

1. Obtain an initial estimate $\hat{\mathbf{U}}, \hat{\mathbf{V}}$ of the switch variables by sampling from $\mathrm{P}(\mathbf{U}, \mathbf{V}|\mathbf{X}, \mathbf{Y})$. Note that this makes use of the ground truth state sequence, $\mathbf{X}$, which is not available during standard test-time inference.

2. Infer $\mathrm{P}(\mathbf{X}|\hat{\mathbf{U}}, \hat{\mathbf{V}}, \mathbf{Y})$, a Gaussian in $\mathbf{X}$, and sample from it a state sequence $\hat{\mathbf{X}}$.

3. Infer $\mathrm{P}(\mathbf{U}, \mathbf{V}|\hat{\mathbf{X}}, \mathbf{Y})$, and from it draw samples of the switches $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$. Note that this time the state sequence sample obtained in Step 2 is used instead of the ground truth $\mathbf{X}$.

4. Goto 2, and repeat this sampling procedure for the desired number of iterations.

To obtain an unbiased sample, the above procedure must be run in theory for an infinite number of iterations, or in practice for enough iterations that it is computationally infeasible. However, Contrastive Divergence notes that a reasonable approximation can be

obtained using a biased sample, obtained after only a very small number of iterations. In practice we use two iterations, and obtain 100 samples to approximate the negative phase expectation.

Computing the gradient for $\boldsymbol{\beta}_k$ can be done analogously. A full derivation of the the learning algorithm is presented in Appendix B.

In is interesting to note that, although Contrastive Divergence is typically used for unsupervised learning, here we apply it to a supervised learning problem. To connect our approach to the more-common unsupervised learning case, the ground-truth state sequences $\mathbf{X}$ can be thought of as the observed data, the switches $\mathbf{U}, \mathbf{V}$ as the hidden variables, and the observations $\mathbf{Y}$ as a set of fixed parameters or biases in an unsupervised model.

3.5 Related Work

Our approach is closely related to previous work on Conditional Random Fields (CRFs) (Lafferty et al., 2001), Products-of-Experts (PoE's) (Hinton, 2002), and energy-based models (Teh et al., 2003), and in some ways can be considered a variant or extension of these frameworks. However, typically CRFs are concerned with discrete states and include simple, discrete features (such as delta-function indicator-variables). In contrast our model works with a combination of continuous and discrete state, incorporates unobserved latent variables during training and testing, and employs continuous-valued feature functions. Some of these elements have been used individually in other models proposed recently, such as (Quattoni et al., 2005; Sudderth et al., 2005), but to our knowledge the particular combination of these elements is novel to the model presented here. Likewise, PoE's or energy-based models have been predominantly applied to unsupervised modelling the full joint density, rather than the conditional posterior as in our case. Within this class of models, our work has some commonalities with exponen-

tial family harmoniums (Welling et al., 2005), where we use Bernoulli/Multinomial and Gaussian layers, with the extension in our model that this distribution is *conditioned* on a set of observation features, and that the Gaussians units have a linear chain dependency. Our work also bears some similarities to the Fields-of-Experts (Roth and Black, 2005), in the sense that we learn an undirected, translation-invariant dependency structure.

Our model draws additional inspiration from several approaches in the general sequential state estimation literature and it shares some commonalities with models based around switching state space models (switching-SSMs) (Ghahramani and Hinton, 2000). As with our work, these generative models employ a set of switches that select between distinct state-transition functions, as well as having switch-dependent emission distributions. Switching-SSMs typically couple the switch states through time in a directed Markov chain; although we currently do not do this, it would be a computationally feasible extension to our model. In relation to switching-SSMs, our model is able to leverage the usual advantages of discriminative training in conditional models — namely that the switch variable can be set up to have a rather rich dependency on the observation sequence without incurring extra difficulties with inference or tractability. Our model differs from switching-SSMs in another important way: SSMs fit the joint probability of state and observations[1], whereas our model disregards the observation density, instead fitting only the conditional probability of state given observations.

With respect to the particular application to object tracking, a number of models employing CRF-style approaches have recently been suggested, including (Sminchisescu et al., 2005; Taycher et al., 2005). While our approach shares some of the same modelling philosophies as these approaches, including employing a variety of features, discriminative training, and dynamic models, the overall form and components of our approach is signif-

[1]When the ground truth state sequence is available during training—the supervised learning case—SSMs are fit to the joint probability of state and observations. In contrast, when only the observation sequence is available—unsupervised learning, for example via expectation-maximization—SSMs are fit to the probability distribution of the observations only.

icantly different. Also from the broader tracking literature there are several approaches that have some aspects in common with our work. In particular, (Collins et al., 2005) shares the notion that it is advantageous to have a candidate pool of mechanisms to estimate the object position, and to swap these in and out based on their local performance and consistency. However, this model has no explicit representation of dynamics, and is restricted to simpler features than those in our framework. In a similar vein, (Forsyth and Ponce, 2002) suggests an approach that employs a Kalman filter in conjunction with "gated" observations. Lastly, we note that (Isard and Blake, 1998) use particle filtering and a switching dynamics model to follow a simple bouncing ball, which influenced our choice of an illustrative experiment using a bouncing ball in a more realistic setting.

3.6 A visual tracking application

As a test of our model, we apply it to the problem of tracking the position of a basketball in video. Here, we show that by combining several different simple (and often unreliable) observation and dynamics features, we can obtain a reliable tracker.

In this setting the observations are a sequence of grayscale images. For the state at time t we use a 6-dimensional vector encoding the position, velocity, and acceleration of the ball. Augmenting the state-space with velocity and acceleration is a standard transformation (Forsyth and Ponce, 2002), allowing higher-order dynamics to be modeled using features that only look at pairs of temporally adjacent states. Training data consists of a sequence of images, as well as the ground truth locations of the target object (with velocity and acceleration computed via finite differencing).

In our tracker, we include eight different observation features. The first six are based on small template images. Given an observation image, each template is compared (efficiently, using convolution) to all possible sub-patches in the image, and $\gamma_k(\mathbf{Y}, t)$ returns the location of the most similar patch based on sum-of-squares distance. The

next feature uses a 3-component principal components analysis (PCA) subspace. Again, the subspace is applied to all areas of the current image, and $\gamma_k()$ returns the location of the image patch with lowest sum-of-squares reconstruction error. The final feature is based on temporally-local background subtraction. It takes five observation images (the current, two proceeding, and two following images), computes the mean image, and returns the point in the current image that differs most from this mean (after Gaussian blur of the difference image). As expected, this feature can work well when there is only one rapidly moving object, but can be very unreliable when there is any other motion (including camera motion) in the image. None of these features are able to estimate the velocity or acceleration, thus for these dimensions of the state the $\phi_k()$'s always predict zero. Note that in our model it is perfectly acceptable for a feature to consider only a subset of the state dimensions.

For each of these observation features we include side-information to help determine the values of the switches. For the template and PCA features, we compute the sum-of-squares error of the best-matching image patch, and for the background subtraction feature, we compute the maximum squared difference between the current and mean images. Each switch potential $\mathcal{G}_k$ takes a (different) linear combination of these values and returns the result. Thus each $\mathcal{G}_k$ can be thought of as a logistic regression classifier, attempting to determine the observation switches v_{kt} using only information from the observation images.

Four dynamics features are included, each using a linear predictor of $\mathbf{x}_t$ from $\mathbf{x}_{t-1}$. We do not include any side-information for the dynamics, thus $\mathcal{F}_j$ is simply a constant bias.

3.6.1 Tracking a basketball

The video data for this evaluation consisted of four video sequences, totalling around 8400 frames, for which the ground truth location of the basketball was obtained by hand.

We trained two separate trackers to evaluate performance during weak and strong generalization scenarios. In the first experiment we used one sequence (referred to as `Simon`, 1796 frames) containing a single player bouncing, throwing and dribbling a basketball. The tracker was trained using the first 500 frames (as well as the corresponding ground truth points) and tested on the remaining frames. In the second experiment we used three sequences in which players pass a basketball by rolling (`roll`, 1556 frames), bouncing (`bounce`, 1897 frames), and by both rolling and bouncing (`roll+bounce`, 3126 frames). Here the tracker was trained on the first 500 frames of `roll` and `bounce`, and tested on the held-out `roll` and `bounce` frames (weak generalization), as well as on the previously unseen `roll+bounce` sequence (strong generalization).

To train the template features, we extracted 19×19-pixel image patches of the basketball from the training images. The first five templates were obtained by running K-means clustering on the patches, while the sixth was simply one of the training patches (we chose the last image). The PCA model was also fit using these training patches. The linear parameters of the switch potentials $\mathcal{G}_k$ were fit using logistic regression. Although they can often correctly locate the basketball, none of the features is always "on the ball". The reliability of each observation feature (the frequency with which it predicts a location within 5 pixels of the basketball) is given in Table 3.2. The most reliable feature is PCA (0.81), and the "background subtraction" feature (0.08) is the least. Included also in Table 3.2 is a measure of how accurately the tracker switches these observation features on or off during the test sequence. As can be seen, switching is very accurate for all features except background subtraction. Closer investigation reveals that although the tracker seems to be erroneously switching this unreliable feature on, in reality it has learned to permanently ignore the feature by assigning it very little weight ($1/35^{th}$ the weight assigned to the PCA feature).

To train the dynamics features ($\mathbf{T}_j$ and $\mathbf{d}_j$), in the first experiment we hand-segmented the ground-truth states from the training data into four regimes: *flying* (the basketball in

Table 3.2: Reliability of the observation features on the `Simon` sequence.

Feature	Fraction of frames in which feature ...	
	correctly locates the ball.	is correctly switched on/off.
K-means 1	0.34	0.98
K-means 2	0.53	1.00
K-means 3	0.61	1.00
K-means 4	0.63	0.96
K-means 5	0.63	1.00
Last training patch	0.33	0.98
PCA	0.81	1.00
Background Subtraction	0.08	0.29

free-flight), *holding* (the basketball in the hands of the player), *bouncing off the ground*, and *bouncing off the wall*. The parameters of each $\phi_j()$ were chosen to minimize $\|\mathbf{x}_t - (\mathbf{T}_j\mathbf{x}_{t-1} + \mathbf{d}_j)\|$ for the set of corresponding ground-truth states. Segmenting the data can be time consuming, so in the second experiment we used manually chosen dynamics features, corresponding to flight, rolling, bouncing, and holding.

Finally, the feature precisions $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_k$, as well as the logistic regression parameters, were refined using 300 iterations of Contrastive Divergence learning. Given the learned model, we track the basketball by applying 20 iterations of the inference method described in Section 3.3, producing an estimate of the state sequence and the switches.

3.6.2 Results

To quantitatively assess the trackers' performance, for each of the test sequences we computed an error rate defined to be the fraction of frames in which the predicted state

was more than 5 pixels away from the true location of the basketball. As a baseline, we also attempted to track the sequences using an SSM, specifically a Kalman filter fit to the training data. We experimented with including different subsets of the observations features , as well as including/not including velocity and acceleration in the state, but in all cases the Kalman filter performed very poorly. For further comparison, we applied the incremental visual tracker (IVT) presented earlier in Chapter 2.

The result of tracking the `Simon` sequence is shown in Figure 3.2. As can be seen the basketball is tracked well throughout the sequence. Although the tracker loses the ball briefly on four occasions, it quickly recovers. The error rate of the tracker was 0.1196, versus 0.7229 for the Kalman filter. The IVT was able to track the first part of the sequence without error, but it lost track of the ball after 688 frames and was unable to recover, resulting in an error rate of 0.613.

The results of the second experiment are shown in Figure 3.3. The error rates for the tracker were 0.0208 on `roll`, 0.0766 on `bounce`, and 0.1004 on `roll+bounce`. The corresponding rates for the Kalman filter were 0.8333, 0.9434, and 0.9697. The IVT tracked the first 700 frames of `roll`, 125 frames of `bounce`, and 730 frames of `roll+bounce` before failing, giving error rates of 0.37, 0.919, and 0.767. The discrepancy in error rates between the three trackers highlights the difficulty in choosing an appropriate metric for quantitatively comparing tracking results. However, we feel that fraction of time "on-the-ball" seems the most appropriate measure for this application.

3.6.3 Dealing with missing observations

To test our model's ability to deal with missing observations, and the quality of the learned dynamics features, we conducted two experiments.

First, we modified the `Simon` sequence so that all observation features would be off for 20 consecutive frames, while the ball is in free flight. Occlusion is a notoriously difficult problem for tracking, as state-of-the-art trackers perform simple diffusion until telltale

observations enable the tracker to locate the object (Jepson et al., 2001). The result can be seen in Figure 3.2 (lower-right image). The model is able to successfully track through the 20-frame (98-pixel displacement) simulated occlusion. Note that the uncertainty in the state estimate (indicated by blue circles of one standard deviation) grows during the missing observations, peaking at the middle of the occlusion.

Second, we applied the tracker trained on the `roll+bounce` sequences to a previously-unseen video in which the basketball passes twice behind a bin, and is completely occluded for approximately 15 frames each time. As can be seen in Figure 3.4, the ball was successfully tracked through both occlusions. Note that this test video is particularly challenging since the ball bounces off the ground while it is occluded. Admittedly, the inferred trajectory during occlusion does not recover the true location of the ball's bounce, however the tracker does posit a plausible trajectory based on its knowledge of the ball's typical dynamics.

3.7 Discussion

We have presented a novel framework for inferring complex trajectories from high-dimensional and noisy data. One of the key advantages of our approach is that we have complete flexibility about the observation and dynamics features that we use in our model. The discriminative learning procedure can appropriately weight the confidence of different predictors, as well as integrating these predictions over time with a versatile dynamics model, and learning to effectively gate in and out different features based on their inferred accuracy and relevance. Although the features used in this work are relatively simple, we still obtain impressive results. One exciting prospect is the possibility of using rather more powerful predictors in combination — for instance one could use as observation features the outputs from state-of-the-art object detectors, or even *other trackers*. We could also use information from multiple frames (for example estimated

optical flow) to help make predictions.

Our model can also readily be extended by improving the side information that constrains the inference of which features to use. Our current model does not utilize side information to help select the dynamics switches; some interactions could be used here to improve performance. Also, the switches are conditionally independent at each time-step. Whilst it might not be practical to couple all the observation switches over time, it seems feasible to take the multinomial switches controlling the dynamics and couple them in a linear Markov chain. Inference (for smoothing) would then consist of a forwards-backwards pass of belief propagation for both the continuous state *and* the dynamics switches, whilst the observation switches would remain conditionally independent given the continuous state.

One of the main drawbacks of our approach is that we require a fully labelled sequence to train on, and in certain applications such data (and in sufficient quantities to constrain all the parameters of the model without encountering problems with overfitting) may be hard to come by. A simple way around this would be to bootstrap from other tracking methods, using them to provide an initial labelling of sequences for training and using human intervention only on the more challenging segments.

In addition to incorporating more powerful features, there are a number of other interesting directions in which this work may be taken. One area that we have started to explore is the performance of different inference algorithms. The dependency structure of our model allows for a relatively efficient MCMC procedure for obtaining samples from the conditional posterior. Other (approximate) inference methods such as variational approximations, particle filtering or non-parametric belief propagation may also be of use in our model, particularly for real-time or online/filtering applications.

Another interesting direction for this work might be the inclusion of multi-modal observations and multi-sensory fusion for inference of state. The features we use need not entirely constrain the state, and one of the strengths of our model is that it is able

to combine many weak predictors to give a single good estimate of state. As a example relative to the experiments presented here, one could imagine a simple auditory cue as being rather useful in inferring the occurrence of a bounce — potentially allowing us to deal with changes in behaviour even when the ball is visually occluded.

Moving beyond the scope of tracking applications considered here, the general ideas behind our model seem to hold promise for a wide range of applications — examples under consideration include articulated body recovery, prediction of financial time series, and flexible combination of stereoscopic depth predictions.

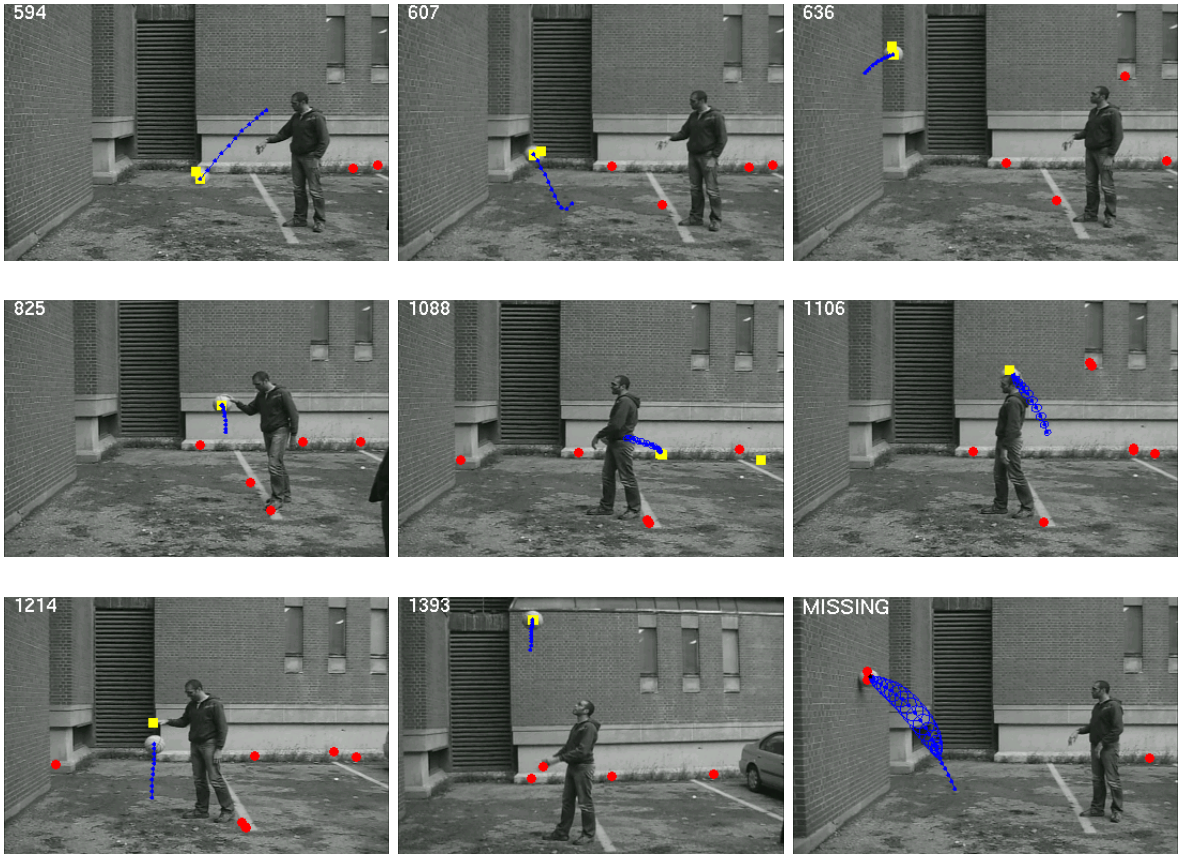

Figure 3.2: Tracking a basketball (`Simon` sequence). The full video is available at http://www.cs.toronto.edu/~dross/phd/. The location of the ball, as predicted by the model, is given in blue. The locations predicted by the eight observations are drawn as yellow boxes (if the corresponding switch is 1) or as red circles (if the switch is 0). The ball is successfully tracked during fast motion (frame 594), when bouncing off the ground (607) and the wall (636), and when dribbled by the player (825). The model does lose track of the ball when it is occluded (1088), but quickly recovers when the ball becomes visible again (1106). The dynamics allow the ball to be tracked even when all observation features' predictions are erroneous (and one bad feature is on!) (1214). The model can also cope with motion of the camera (1393). (MISSING DATA: lower right) The basketball is successfully tracked through a 20-frame (98 pixel displacement) simulated occlusion.

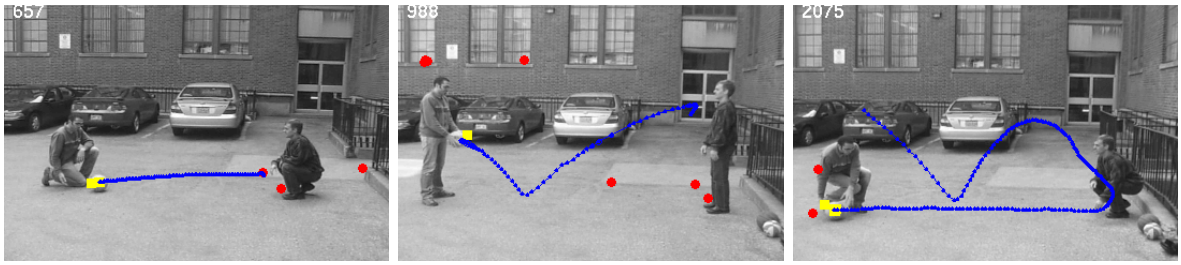

Figure 3.3: Results on the `roll`, `bounce`, and `roll+bounce` sequences. The full videos are available at http://www.cs.toronto.edu/~dross/phd/.

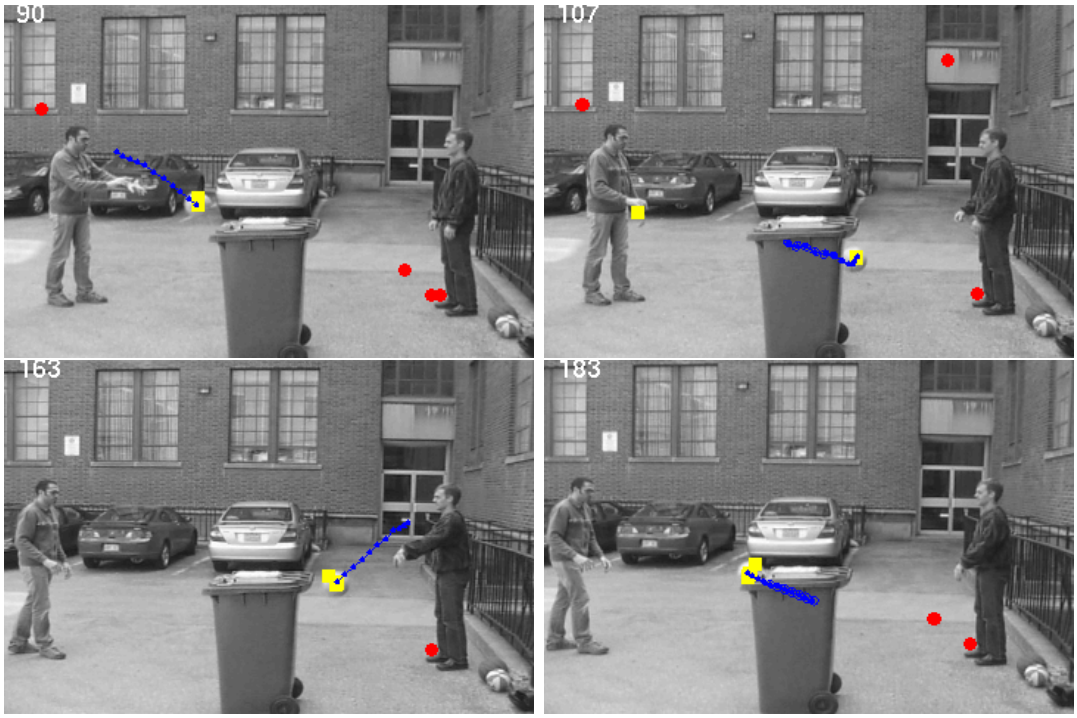

Figure 3.4: The basketball is successfully tracked as it passes behind a bin, first when it is thrown from left to right (frames 90–107), and then a second time as is is thrown back from right to left (frames 163–183).

Chapter 4

Learning Articulated Structure and Motion

Humans demonstrate a remarkable ability to parse complicated motion sequences into their constituent structures and motions. We investigate this problem, attempting to learn the structure of one or more articulated objects, given a time series of two-dimensional feature positions. We model the observed sequence in terms of "stick figure" objects, under the assumption that the relative joint angles between sticks can change over time, but their lengths and connectivities are fixed. The problem is formulated as a single probabilistic model that includes multiple sub-components: associating the features with particular sticks, determining the proper number of sticks, and finding which sticks are physically joined. We test the algorithm on challenging datasets of 2D projections of optical human motion capture and feature trajectories from real videos.

4.1 Introduction

An important aspect of analyzing dynamic scenes involves segmenting the scene into separate moving objects and constructing detailed models of each object's motion. For scenes represented by trajectories of features on the objects, structure-from-motion meth-

ods are capable of grouping the features and inferring the object poses when the features belong to multiple independently moving rigid objects. Recently, however, research has been increasingly devoted to more complicated versions of this problem, when the moving objects are articulated and non-rigid.

In this chapter we investigate the problem, attempting to learn the structure of an articulated object while simultaneously inferring its pose at each frame of the sequence, given a time series of feature positions. We propose a single probabilistic model for describing the observed sequence in terms of one or more "stick figure" objects. We define a "stick figure" as a collection of line segments (bones or sticks) joined at their endpoints. The structure of a stick figure—the number and lengths of the component sticks, the association of each feature point with exactly one stick, and the connectivity of the sticks—is assumed to be temporally invariant, while the angles (at joints) between the sticks are allowed to change over time. We begin with no information about the figures in a sequence, as the model parameters and structure are all learned. An example of a stick figure learned by applying our model to 2D feature observations from a video of a walking giraffe is shown in Figure 4.1.

Learned models of skeletal structure have many possible uses. For example, detailed, manually constructed skeletal models are often a key component in full-body tracking algorithms. The ability to learn skeletal structure could help to automate the process, potentially producing models more flexible and accurate than those constructed manually. Additionally, skeletons are necessary for converting feature point positions into joint angles, a standard way to encode motion for animation. Furthermore, knowledge of the skeleton can be used to improve the reliability of optical motion capture, permitting disambiguation of marker correspondence and occlusion (Herda et al., 2001). Finally, a learned skeleton might be used as a rough prior on shape to help guide image segmentation (Bray et al., 2006).

In the following section we discuss other recent approaches to modelling articulated

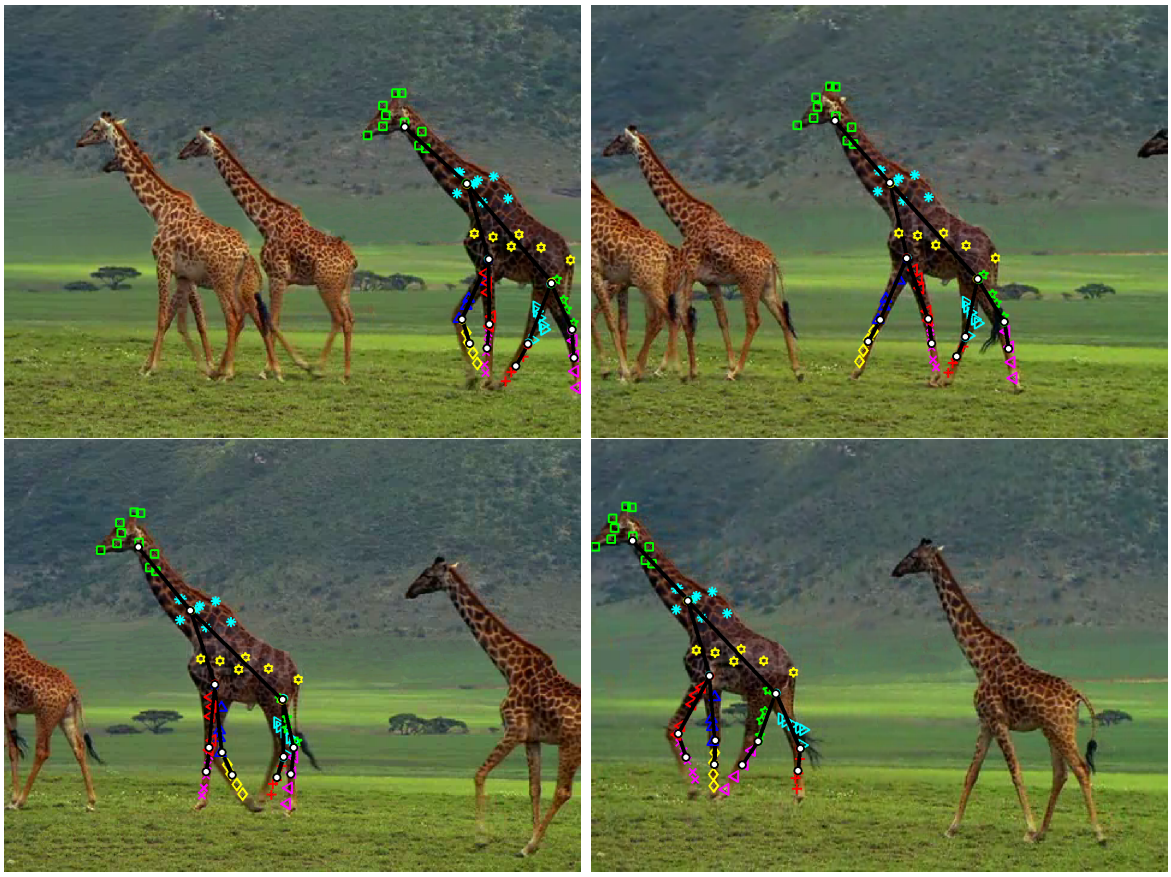

Figure 4.1: Four frames from a video of a walking giraffe, with the articulated skeleton learned by our model superimposed. Each black line represents a stick, and each white circle a joint between sticks. The tracked features, which serve as the only input, are shown as coloured markers. Features associated with the same stick are assigned markers of the same colour and shape.

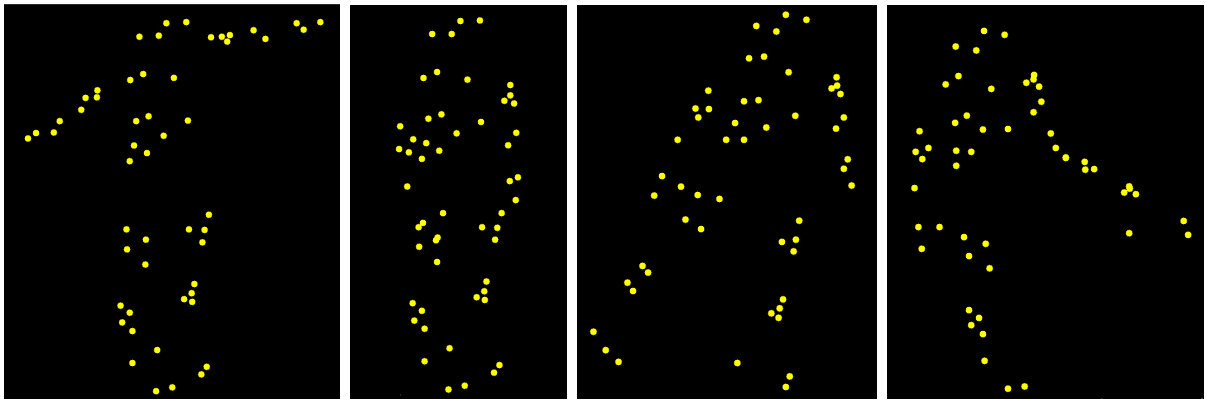

Figure 4.2: A point light display of a human, in four different poses.

figures from tracked feature points. In Section 4.3 we formulate the problem as a probabilistic model, and in Section 4.4 we propose an algorithm for learning the model from data. Learning proceeds in a stage-wise fashion, building up the structure incrementally to maximize the joint probability of the model variables.

In Section 4.5 we test the algorithm on a range of datasets. In the final section we describe assumptions and limitations of the approach, and discuss future work.

4.2 Related Work

Humans demonstrate a remarkable ability to parse complicated motion sequences, even from apparently sparse streams of information. One field where this is readily apparent is in the study of human response to point light displays. A point light display (PLD), as depicted in Figure 4.2, is constructed by attaching a number of point light sources to an object, then recording (only) the positions of these lights as the object moves. The canonical example is to instrument a human's limbs and body with lights, then to record their positions as he or she performs motions such as walking, running, or swinging a golf club. PLDs have received considerable attention in psychology research (*e.g.* Johansson, 1973) due to one remarkable property. Despite the apparently limited information they

contain, biological motion depicted in PLDs is almost instantly recognizable by humans. From a PLD of a person or animal, humans are able to understand the structure of the display (how the lights are connected via the performer's underlying skeleton), and the motions that are performed.

Point light displays are also common in several domains of computer science research. The field of motion capture, in essence, is the study of recording and analyzing PLDs. In computer animation, PLDs obtained via motion capture are used to animate synthetic character models. Finally, in computer vision many applications choose to represent digital images sequences in terms of feature point trajectories. When the original image data is discarded, the feature points locations are equivalent to a PLD.

What follows is a discussion of three recent approaches to modelling articulated figures from tracked feature points. Each of these approaches addresses the problem from a different viewpoint: the first as structure from motion, the second as geometrical constraints in motion capture data, and the third as learning the structure of a probabilistic graphical model.

4.2.1 Articulated Structure From Motion

The first work we will consider is "Automatic Kinematic Chain Building from Feature Trajectories of Articulated Objects" by Yan and Pollefeys (2006b). The premise of this work is to extend standard solutions for the *structure from motion* (SFM) problem to handle articulated objects. Given a set of feature points observed at a number of frames, the goal of SFM is to recover the *structure*—the time-invariant relative 3D positions of the points—while simultaneously solving for the *motion*—the per-frame pose of the object(s) relative to the camera—that produced the observations. Generally, the input for SFM is assumed to be two-dimensional observations (image coordinates) of points on an inherently three-dimensional object. However most algorithms, including the ones presented here, work equally well given 3D inputs.

When the trajectories come from one rigid object (or equivalently, the scene is static and only the camera moves), Tomasi and Kanade (1992) have shown that structure and motion can recovered by using the singular value decomposition (SVD) to obtain a low-rank factorization of the matrix of feature point trajectories.

Suppose we are given a matrix $\mathbf{W}$ where each column contains the x and y image coordinates of one of the observed points, at all time frames. Thus, given P points and F frames, the size of $\mathbf{W}$ is $2F \times P$ (or $3F \times P$ for three-dimensional observations). Considering the generative process that produced the observations (and disregarding noise), $\mathbf{W}$ is the product of a motion matrix and a structure matrix,

$$\mathbf{W} = \mathbf{MS},$$

both of which are rank 4. The structure $\mathbf{S}$ is a $4 \times P$ matrix containing the time-invariant (homogeneous) 3D coordinates of the points. At each frame f, the observations are produced by applying a rigid-body motion—a rotation $\mathbf{R}_f$ and a translation $\mathbf{t}_f$—to $\mathbf{S}$, and projecting the points onto the image plane:

$$\begin{bmatrix} x_{f,1} & \dots & x_{f,P} \\ y_{f,1} & \dots & y_{f,P} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \end{bmatrix} \mathbf{S}.$$

Hence, $\mathbf{M}$ is formed by stacking the first two rows of each of these F motion matrices. From $\mathbf{W}$, $\mathbf{M}$ and $\mathbf{S}$ can be recovered by taking the singular value decomposition[1]:

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \quad \Rightarrow \quad \mathbf{M} = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}} \qquad \mathbf{S} = \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^\top.$$

In practice, feature trajectories will be contaminated by noise, giving $\mathbf{W}$ a rank larger than 4. In this case Tomasi and Kanade suggest retaining only the columns of $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V}$ corresponding to the four largest singular values, which is the optimal rank-4 approximation to W (under squared error).

[1] In most cases, although the columns of $\mathbf{U}$ and $\mathbf{V}$ span the correct subspaces, they are actually linear transformations of the columns of $\mathbf{M}$ and $\mathbf{S}$ respectively. That is $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{A}$ and $\mathbf{S} = \mathbf{A}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^\top$, for some unknown invertible $\mathbf{A}$. This can be corrected by solving, via nonlinear optimization, for the $\mathbf{A}$ that satisfies the constraints on the rotational components of $\mathbf{M}$: $\mathbf{M}_{i,1:3}{\mathbf{M}_{i,1:3}}^\top = 1$ and $\mathbf{M}_{i,1:3}{\mathbf{M}_{i+1,1:3}}^\top = 0$

Despite the elegance and popularity of this solution, Tomasi and Kanade (1992) assume a rather unrealistic camera model—scaled orthography—for the projection of three-dimensional points down to two dimensions. As such, this does not represent a complete solution to rigid-body SFM.[2] However, when the input consists of three-dimensional points (*e.g.* obtained from a motion capture system), scaled orthography is perfectly reasonable assumption.

Multibody SFM

Recovering structure and motion when the scene contains multiple objects moving independently is more challenging. Consider the case in which the point trajectories arise from two independent rigid objects. If the columns of $\mathbf{W}$ are sorted so that all points from object 1 come first, and the points from object 2 come second, the low-rank factorization can be written as follows:

$$\mathbf{W} = \mathbf{MS} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & 0 \\ 0 & \mathbf{S}_2 \end{bmatrix}. \tag{4.1}$$

In this case the ranks of the motion and structure matrices (and hence, of $\mathbf{W}$) have increased to 8, or 4× the number of objects. If the grouping of point trajectories into objects was known, the structure and motion of each object, $\mathbf{M}_i$ and $\mathbf{S}_i$, could be recovered independently, using the method described earlier. The problem now becomes, how to group the points?

The solution proposed by Costeira and Kanade (1996, 1998) involves considering what they term the *shape-interaction matrix*, $\mathbf{Q} \equiv \mathbf{V}\mathbf{V}^\top$. When the columns of $\mathbf{W}$ are correctly

[2]Solutions based on the more-realistic projective camera, perhaps using the above method as an initialization, can be obtained via an algorithm for *bundle adjustment* (Hartley and Zisserman, 2003).

sorted, as in (4.1), $\mathbf{Q}$ assumes a distinctive block-diagonal structure[3]

$$\mathbf{Q} \equiv \mathbf{V}\mathbf{V}^\top = \mathbf{S}^\top \mathbf{\Sigma}^{-1} \mathbf{S} = \begin{bmatrix} \mathbf{S}_1{}^\top \mathbf{\Sigma}_1^{-1} \mathbf{S}_1 & 0 \\ 0 & \mathbf{S}_2{}^\top \mathbf{\Sigma}_2^{-1} \mathbf{S}_2 \end{bmatrix},$$

where $\mathbf{V}$ and $\mathbf{\Sigma}$ again arise from the SVD of $\mathbf{W}$. Regardless of the sorting of the points, $\mathbf{Q}_{i,j}$ is nonzero if points i and j are part of the same rigid object, and 0 otherwise. The shape-interaction matrix has the advantage of being invariant to object motion, image scale, and choice of coordinate system.

Costeira and Kanade suggest that grouping point trajectories can now be accomplished by reordering the points to make $\mathbf{Q}$ block-diagonal. This problem, however, is NP-complete, thus the greedy algorithm they propose obtains only sub-optimal solutions. Interestingly, $\mathbf{Q}$ can be interpreted as a pairwise affinity matrix. In fact, $\mathbf{V}\mathbf{V}^\top$ is simply a weighted version of the inner product matrix $\mathbf{W}^\top\mathbf{W}$. This interpretation suggests that other ways of normalizing the shape-interaction matrix are possible, and that points could be grouped by any clustering algorithm which takes as input an affinity matrix, such as spectral clustering (Shi and Malik, 2000; Culverhouse and Wang, 2003; Weiss, 1999).

The primary disadvantage to this approach is that the shape-interaction matrix is highly sensitive to noise in the observations (Gruber and Weiss, 2004). First of all, in the presence of noise $\mathbf{Q}_{i,j}$ is no longer zero when i and j come from different objects. Furthermore, computing $\mathbf{Q}$ requires knowing the rank of $\mathbf{W}$, which is the number of columns of $\mathbf{V}$ retained after the SVD. (Note that if we retain all columns of $\mathbf{V}$, then $\mathbf{Q} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$.) In the simplest case, this rank is $4\times$ the number of objects, but it can be less when an object does not express all its degrees of mobility. Noise makes the rank of $\mathbf{W}$ difficult to determine, requiring an often-unreliable analysis of the eigenspectrum.

[3] $\mathbf{V}\mathbf{V}^\top$ is also block-diagonal if we allow $\mathbf{V}^\top$ to more generally be an invertible linear transformation of the true structure: $\mathbf{S} = \mathbf{A}^{-1}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^\top$ (Costeira and Kanade, 1998).

Probabilistic SFM

Gruber and Weiss (2003) have noted that the approach of Tomasi and Kanade can be reinterpreted as a probabilistic graphical model, specifically *factor analysis*. In factor analysis, each observed data vector is generated by taking a linear combination of a set of basis vectors, and adding diagonal-covariance Gaussian noise. In the context of single-body SFM each row $\mathbf{w}_i$ of $\mathbf{W}$, the x or y coordinates of all feature points in one frame, is generated by taking a linear combination $\mathbf{m}_i$ of the rows of $\mathbf{S}$. Including a standard Gaussian prior on the rows of the motion matrix produced the following model:

$$\begin{aligned} \mathbf{w}_i &= \mathbf{m}_i\mathbf{S} + \mathbf{n}_i, \quad \text{where} \\ \mathbf{n}_i &\sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\psi}_i)) \\ \mathbf{m}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned}$$

Structure and motion can be recovered by fitting the model using the standard Expectation-Maximization (EM) algorithm for factor analysis (Ghahramani and Hinton, 1996a). An advantage of this formulation is that missing observations can be dealt with easily; setting the corresponding variances to ∞ has the effect of eliminating them from the calculations (Gruber and Weiss, 2003).

Another key innovation of the Gruber and Weiss approach is to assume temporal coherence of motions. This allows them to take advantage of the fact, when estimating motions, that motions for adjacent frames should be similar. In the graphical model, temporal coherence is incorporated easily through the use of a Markov chain prior (a Kalman filter) over the latent motion variables. The result is closely related to the EM algorithm for learning linear dynamical systems (Ghahramani and Hinton, 1996b).

Multibody factorization

The probabilistic approach has also been extended to handle multiple independent rigid objects (Gruber and Weiss, 2004). Structure and motion are modeled in much the same

way as (Costeira and Kanade, 1996): one independent factor analyzer of dimension 4 for each object. However, the approach of Gruber and Weiss to grouping point trajectories is quite different.

Instead of grouping points by clustering a pairwise affinity matrix, Gruber and Weiss incorporate additional discrete latent variables that assign each of the points to one of the motions. With this addition, the grouping, together with the structures and motions, can be estimated jointly using EM. This provides a distinct advantage over the method of Costeira and Kanade which, once it has grouped the points, is unable to reestimate the grouping based on subsequent information. Although fitting with EM often leads to local minima, in the presence of noise it outperforms Costeira and Kanade.

The core of this model is the same as *Multiple Cause Factor Analysis* (Ross and Zemel, 2006), independently proposed for simultaneous segmentation and appearance modelling of images.

Articulated Structures

The motion of an articulated object can be described as a collection of rigid motions, one per part, with the added constraint that the motions of connected parts must be spatially coherent. Yan and Pollefeys (2005a) have shown that this constraint causes the motion subspaces of two connected objects to intersect, making them linearly dependent. In particular, for each pair of connected parts, the motion subspaces share one dimension (translation) if they are joined at a point and two dimensions (translation and one angle of rotation) if they are joined at an axis of rotation. As a result of this dependence, the method of Costeira and Kanade (1996) for grouping points is no longer applicable.

To illustrate this, consider two parts that are connected by a rotational joint. Without loss of generality the shape matrices of the objects, $\mathbf{S}_1$ and $\mathbf{S}_2$ (dropping the homogeneous coordinate) can be adjusted to place this joint at the origin. Now, because the objects are connected at the joint, at each frame the translation components of their motions

must be identical. Thus the ranks of $\mathbf{W}$, $\mathbf{M}$, and $\mathbf{S}$ have been reduced to at most 7 (Yan and Pollefeys, 2005a,b).

$$\mathbf{W} = \mathbf{MS} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \\ \mathbf{1} & \mathbf{1} \end{bmatrix}$$

From this equation, we can see that the off-diagonal blocks of the shape interaction matrix, $\mathbf{V}\mathbf{V}^\top = \mathbf{S}^\top \mathbf{\Sigma}^{-1} \mathbf{S}$, are no longer zero, so clustering it will not effect the grouping of point trajectories.

Recognizing this, Yan and Pollefeys (2006a,b) propose an alternative affinity matrix to use for grouping points, and an approach for recovering the full articulated structure and motion of the sequence. Their method consists of four key steps: (1) segmenting the feature point trajectories into a number of rigid parts, (2) computing an affinity measure indicating the likelihood that each pair of parts is connected by a joint, (3) obtaining a spanning tree that connects parts while maximizing affinity, and finally (4) solving for the locations of joints.

When specifying the affinity between a pair of features, instead of relying on the dot product (angle) between rows $\mathbf{v}_i$ and $\mathbf{v}_j$ of $\mathbf{V}$, they suggest that a more robust measure could be obtained by comparing the subspace spanned by $\mathbf{v}_i$ and its nearest neighbors with that of $\mathbf{v}_j$ and its neighbors. Given these two subspaces, they compute the principal angles $\theta_1, \ldots, \theta_m$ between them, and define the affinity between i and j to be

$$\exp\left(-\sum_n \sin^2(\theta_n) \right).$$

The affinity is used as input for spectral clustering (Shi and Malik, 2000), thereby producing a grouping of feature point trajectories.

Principal angles are also used as a basis for learning the articulated structure. Noting that the four-dimensional motions (and hence shape subspaces) of parts connected by an articulated joint will have at least one dimension in common, at least one of the principal

angles between the parts should be zero. Using minimum principal angle as an edge weight, Yan and Pollefeys set up a fully connected graph and solve for the articulated structure by finding the minimum spanning tree. The method can be extend this to finding multiple articulated objects in a scene simply by disallowing edges with weight exceeding a manually specified threshold.

Finally, the locations of the joints can be obtained from the intersections of the motion subspaces of connected parts, as described in (Yan and Pollefeys, 2005a)

Due to the reliance on estimating subspaces, this method requires each body part to have at least as many feature points as the dimensionality of its motion subspace. (In practice, segmenting two independent objects requires at least five points per object, using at least three neighbors to estimate the local subspace, in the noise-free case.)

One drawback of this algorithm is its sequential nature. Each step requires the estimation of a number of quantities: the rank of motion subspaces, the choice of neighbours, the number of clusters, the grouping of points, etc. Each of these choices must be made without access to information obtained from later stages of processing, and since there is no provision for reestimating the values, any mistake is likely to result in poor overall performance.

4.2.2 Geometric Analysis of Motion Capture Data

When observations are the 3D world locations of feature points, rather than 2D projections, the geometry of recovering 3D skeletal structure becomes easier. Based on a simple analysis of the distance between feature points, and following roughly the same four steps as Yan and Pollefeys (2006b), Kirk et al. (2005) are able to automatically recover skeletal structure from motion capture data. This is an improvement upon existing methods of fitting a skeleton to motion capture data (*e.g.* Silaghi et al., 1998; Abdel-Malek et al., 2004), which often require a user to manually associate markers with positions on a generic human skeleton.

The key property motivating the approach of Kirk et al. (2005) is, if two feature points are attached to the same rigid body part, then the distance between these points is constant. Furthermore, if two body parts are connected by a rotational joint, then the distances between the joint and the points belonging to both parts should also be constant. Feature points are grouped, to obtain body parts, by computing the standard deviation of the distance between each pair of points and using that as the (negative or inverse) affinity matrix for spectral clustering (Ng et al., 2002). The number of body parts is chosen manually, or again by analysis of the eigenspectrum.

When determining the skeletal connectivity of the body parts, Kirk et al. define a *joint cost*, which is the average variance in the distance from a putative joint to each of the points in the two parts it connects. Joint costs are computed for each pair of body parts. Evaluating the joint cost requires non-linear conjugate gradient minimization, but also returns the optimal joint location at each frame. Note that joint locations can be estimated as long as one stick has at least two observed markers and the other stick has at least one. Finally, the skeletal structure is obtained by running a minimum spanning tree algorithm, using the joint costs as edge weights.

This method has a few drawbacks. First, it is only able to work on 3D observations– none of the distance constraints it relies upon apply when points are projected into 2D. Second, like (Yan and Pollefeys, 2006b), it consists of a sequence of steps without feedback or reestimation. Finally, beyond computing the positions of joints in each frame, the method does not produce a time-invariant model of structure or a set of motion parameters. As such, filling in missing observations or computing joint angles would require further processing.

One further caveat regarding this method is that, contrary to the images included in the paper (Kirk et al., 2005), its output is not actually a "stick figure"—a collection of line segments (bones or sticks) joined at their endpoints. Instead, in the learned graph, parts of the body are nodes and joints are edges, which is a more-difficult structure to

visualize.

4.2.3 Learning a Graphical Model Structure

Another approach to the analysis of PLDs is to model the relationships between feature point locations with a probabilistic graphical model. In this setting, recovering the skeleton is a matter of learning the graph structure and parameters of the model. This is the approach taken by Song et al. (2001, 2003), with a goal of automatically detecting human motion in cluttered scenes.

Treating each frame as an independent, identically distributed sample, Song et al. construct a model in which each variable node represents the position and velocity of one of the observed points. No latent variables are included, instead each feature point is treated as a unique part of the body. This presumes a much sparser set of features than (Yan and Pollefeys, 2006b) and (Kirk et al., 2005), which require each part to give rise to multiple feature point trajectories. The set of graphs considered is restricted to a particular class, *decomposable triangulated graphs*, in which all cliques are of size three. The limitation placed on the structure ensures that, although these graphs are more complicated than trees, efficient exact inference is still possible. The clique potentials, over triplets of nodes, are multivariate Gaussian distributions over the velocities and relative positions of the parts.

The maximum likelihood (ML) graph is the one that minimizes the empirical entropy of each feature point given its parents. Unfortunately no tractable algorithm exists for computing the ML graph, so Song et al. propose the following approximate greedy algorithm. Assuming all nodes are initially disconnected, choose the first edge in the graph by connecting the nodes B and C that minimize the joint entropy $h(B, C)$. Then, for all possible ways of choosing an pair of connected parents (B, C) already in the graph, find the child A that minimizes the conditional entropy $h(A|B, C)$ and connect it to the graph. Continue connecting child nodes to the graph until it has reached the desired size,

or the entropy of the best putative child exceeds a threshold. The cost of this algorithm is $O(n^4)$, where n is the number of feature points.

Note that if the class of graphical models considered is restricted to trees, the graph structure can be found efficiently, by calculating the mutual information between each pair of body parts and solving for the maximum spanning tree (Taycher et al., 2002; Song et al., 2003).

Song et al. further extend their approach to handle cluttered scenes, obtained by automatically tracking features in video. Since the results of tracking are invariably noisy, this requires solving the correspondence problem at each frame (identifying which feature points are body parts, which come from the background, and which body parts are occluded). Learning can now be accomplished via an EM-like algorithm, which alternates optimizing the feature correspondence with learning the graphical model structure and parameters.

Although the authors are able to show some interesting results, this approach has a number of drawbacks. First, learned models are specific to the 3D position and orientation of the subject, accounting only for invariance to translation parallel to the image plane. Thus a model trained on a person walking from left to right is unable to detect a person walking from right to left (Song et al., 2003). Secondly, a single time-invariant model is learned on the data from all frames, thereby confounding structure and motion. Instead of trying to model these two latent factors separately, the presence of motion serves only to increase uncertainty in the graphical model.

4.3 Model

Here we formulate a probabilistic graphical model for sequences generated from articulated skeletons. By fitting this model to a set of feature point trajectories (the observed locations of a set of features across time), we are able to parse the sequence into one

or more articulated skeletons and recover the corresponding motion parameters for each frame. The observations are assumed to be 2D, whether tracked from video or projected from 3D motion capture, and the goal is to learn skeletons that capture the full 3D structure. Fitting the model is performed entirely via unsupervised learning; the only inputs are the observed trajectories, with manually tuned parameters restricted to a small set of thresholds on Gaussian variances.

The observations for this model are the locations $\mathbf{w}_p^f$ of feature points p in frames f. A discrete latent variable $\mathbf{R}$ assigns each point to one of S sticks. Each stick s consists of a set of time-invariant 3D local coordinates $\mathbf{L}_s$, describing the relative positions of all points belonging to the stick. $\mathbf{L}_s$ is mapped to the observed world coordinate system by a different motion matrix $\mathbf{M}_s^f$ at every frame f (see Figure 4.3). For example, in a noiseless system, where $r_{p,1} = 1$, indicating that point p has been assigned to stick 1, $\mathbf{M}_1^f \mathbf{l}_{1,p} = \mathbf{w}_p^f$.

If all of the sticks are unconnected and move independently, then this model essentially describes multibody SFM (Costeira and Kanade, 1998; Gruber and Weiss, 2004), or equivalently an instance of *Multiple Cause Factor Analysis* (Ross and Zemel, 2006). However, for an articulated structure, with connections between sticks, the stick motion variables are not independent (Yan and Pollefeys, 2006a). Allowing connectivity between sticks makes the problems of describing the constraints between motions and inferring motions from the observations considerably more difficult.

To deal with this complexity, we introduce variables to model the connectivity between sticks, and the (unobserved) locations of stick endpoints and joints in each frame. Every stick has two endpoints, each of which is assigned to exactly one *vertex*. Each vertex can correspond to one or more stick endpoints (vertices assigned two or more endpoints are joints). We will let $\mathbf{k}_i$ specify the coordinates of endpoint i relative to the local coordinate system of its stick, $s(i)$, and $\mathbf{v}_j^f$ and $\mathbf{e}_i^f$ represent the world coordinate location of vertex j and endpoint i in frame f, respectively. Again, in a noiseless system, $\mathbf{e}_i^f = \mathbf{M}_{s(i)}^f \mathbf{k}_i$ for

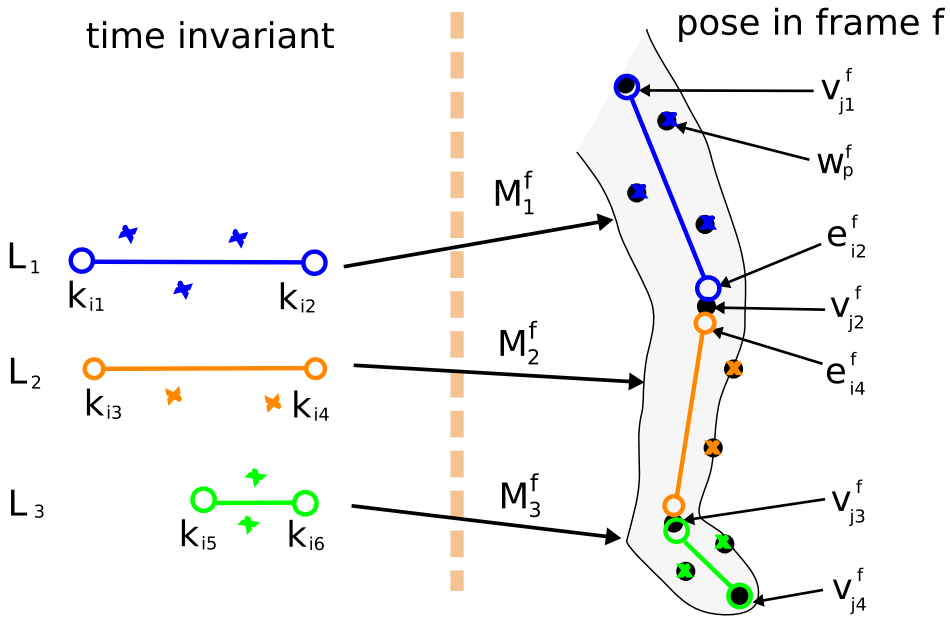

Figure 4.3: The generative process for the observed feature positions, and the imputed positions of the stick endpoints. For each stick, the relative positions of its feature points and endpoints are represented in a time-invariant local coordinate system (left). For each frame in the sequence (right), motion variables attempt to fit the observed feature positions (*e.g.* $\mathbf{w}_P^f$) by mapping local coordinates to world coordinates, while maintaining structural cohesion by mapping stick endpoints to inferred vertex (joint) locations.

every frame f. Noting the similarity between the $\mathbf{e}_i^f$ variables and the observed feature positions $\mathbf{w}_p^f$, these endpoint locations can be interpreted as a set of pseudo-observations, inferred from the data rather than directly observed.

Vertices are used to enforce a key constraint: for all the sticks that share a given vertex, the motion matrices should map their local endpoint locations to a consistent world coordinate. This restricts the range of possible motions to only those resulting in appropriately connected figures. For example, in Figure 4.3, endpoint 2 (of stick 1), is connected to endpoint 4 (of stick 2); both are assigned to vertex 2. Thus in every frame f both endpoints should map to the same world location, the location of the knee joint, *i.e.* $\mathbf{v}_2^f = \mathbf{e}_2^f = \mathbf{e}_4^f$.

The utility of introducing these additional variables is that, given the vertices $\mathbf{V}$ and endpoints $\mathbf{E}$, the problem of estimating the motions and local geometries ($\mathbf{M}$ and $\mathbf{L}$) factorizes into S independent structure-from-motion problems, one for each stick. Latent variable $\mathbf{g}_{i,j} = 1$ indicates that endpoint i is assigned to vertex j; hence $\mathbf{G}$ indirectly describes the connectivity between sticks. The assumed generative process for the feature observations and the vertex and endpoint pseudo-observations is shown in Figure 4.3, and the corresponding probabilistic model is shown in Figure 4.4.

The complete joint probability of the model can be decomposed into a product of two likelihood terms, one for the true feature observations $\mathbf{W}$ and the second for the endpoint pseudo-observations $\mathbf{E}$, and priors over the remaining variables in the model:

$$\begin{aligned}\mathbb{P} = \mathrm{P}(\mathbf{W}|\mathbf{M},\mathbf{L},\mathbf{R})\mathrm{P}(\mathbf{E}|\mathbf{M},\mathbf{K},\mathbf{V},\boldsymbol{\phi},\mathbf{G}) \\ \mathrm{P}(\mathbf{V})\mathrm{P}(\boldsymbol{\phi})\mathrm{P}(\mathbf{M})\mathrm{P}(\mathbf{L})\mathrm{P}(\mathbf{K})\mathrm{P}(\mathbf{R})\mathrm{P}(\mathbf{G})\end{aligned} \tag{4.2}$$

Assuming isotropic Gaussian noise with precision (inverse variance) τ_w, the likelihood function is

$$\mathrm{P}(\mathbf{W}|\mathbf{M},\mathbf{L},\mathbf{R}) = \prod_{f,p,s} \mathcal{N}(\mathbf{w}_p^f|\mathbf{M}_s^f\mathbf{l}_{s,p}, \tau_w^{-1}\mathbf{I})^{r_{p,s}} \tag{4.3}$$

where $r_{p,s}$ is a binary variable equal to 1 if and only if point p has been assigned to stick

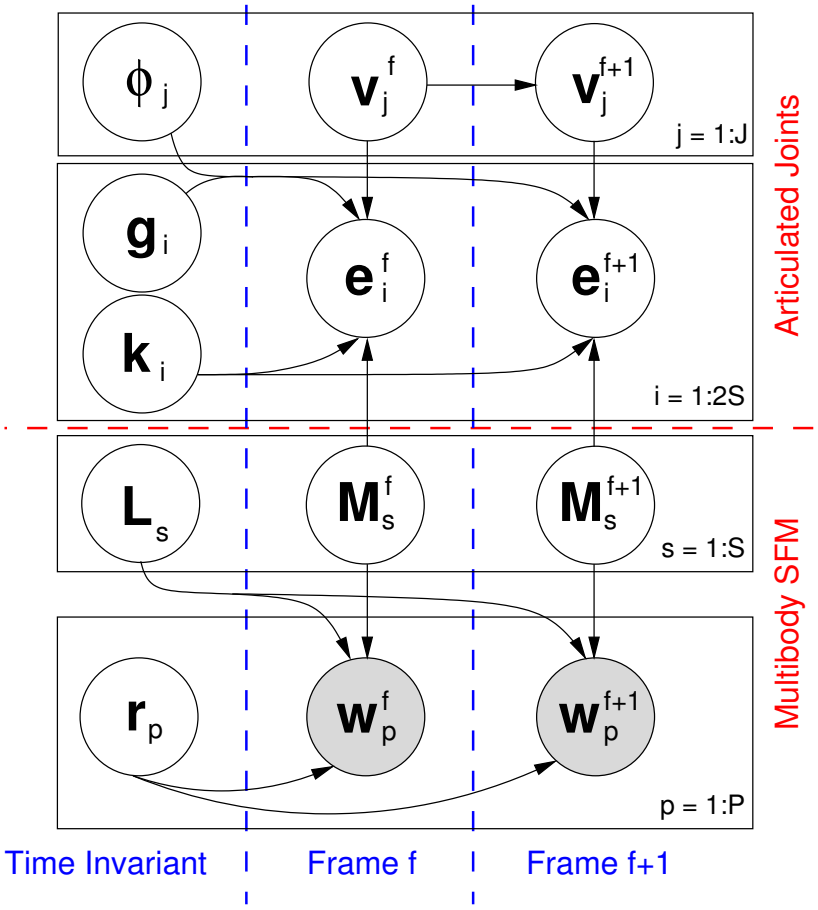

Figure 4.4: The graphical model. The bottom half shows the model for independent multibody SFM; the top half describes the vertices $\mathbf{v}$ and endpoints $\mathbf{e}$, which account for motion dependencies introduced by the articulated joints.

s. This distribution captures the constraint that for feature point p, its predicted world location, based on the motion matrix and its location in the local coordinate system for the stick to which it belongs ($r_{p,s} = 1$), should match its observed world location. Note that dealing with missing observations is simply a matter of removing the corresponding factors from this likelihood expression.[4]

Each motion variable consists of a 2×3 rotation matrix $\mathbf{R}_s^f$ and a 2×1 translation vector $\mathbf{t}_s^f$: $\mathbf{M}_s^f \equiv [\mathbf{R}_s^f \quad \mathbf{t}_s^f]$. The motion prior $\mathrm{P}(\mathbf{M})$ is uniform, with the stipulation that all rotations be orthogonal: $\mathbf{R}_s^f \mathbf{R}_s^{f\top} = \mathbf{I}$.

We define the missing-data likelihood of an endpoint location as the product of two Gaussians, based on the predictions of the appropriate vertex and stick:

$$\mathrm{P}(\mathbf{E}|\mathbf{M}, \mathbf{K}, \mathbf{V}, \boldsymbol{\phi}, \mathbf{G}) \propto \prod_{f,i} \mathcal{N}(\mathbf{e}_i^f | \mathbf{M}_{s(i)}^f \mathbf{k}_i, \tau_m^{-1}\mathbf{I}) \prod_{f,i,j} \mathcal{N}(\mathbf{e}_i^f | \mathbf{v}_j^f, \phi_j^{-1}\mathbf{I})^{g_{i,j}} \tag{4.4}$$

Here τ_m is the precision of the isotropic Gaussian noise on the endpoint locations with respect to the stick, and $g_{i,j}$ is a binary variable equal to 1 if and only if endpoint i has been assigned to vertex j. The second Gaussian in this product captures the requirement that endpoints belonging to the same vertex should be coincident. Instead of making this a hard constraint, connectivity is softly enforced, allowing the model to accommodate a certain degree of non-rigidity in the underlying structure, as illustrated by the mismatch between endpoint and vertex positions in Figure 4.3. The vertex precision variables ϕ_j capture the degree of "play" in the joints, and are assigned Gamma prior distributions:

$$\mathrm{P}(\boldsymbol{\phi}) = \prod_j \mathrm{Gamma}(\phi_j | \alpha_j, \beta_j). \tag{4.5}$$

The prior on the vertex locations incorporates a temporal smoothness constraint, with

[4]This likelihood is applicable if the observations $\mathbf{w}_p^f$ are 2D or 3D. In the 2D case, we assume an affine camera projection. However, it would be possible to extend this to a projective camera by making the mean depend non-linearly on $\mathbf{M}_s^f \mathbf{l}_{s,p}$.

precision τ_t:

$$\mathrm{P}(\mathbf{V}) = \prod_{f,j} \mathcal{N}(\mathbf{v}_j^f | \mathbf{v}_j^{f-1}, \tau_t^{-1}\mathbf{I}) \tag{4.6}$$

The priors for feature and endpoint locations in the local coordinate frames, $\mathbf{L}$ and $\mathbf{K}$, are zero-mean Gaussians, with isotropic precision τ_p.

$$\mathrm{P}(\mathbf{L}) = \prod_{s,p} \mathcal{N}(\mathbf{l}_{s,p} | 0, \tau_p^{-1}\mathbf{I}) \qquad \mathrm{P}(\mathbf{K}) = \prod_{i} \mathcal{N}(\mathbf{k}_i | 0, \tau_p^{-1}\mathbf{I})$$

Finally, the priors for the variables defining the structure of the skeleton, $\mathbf{R}$ and $\mathbf{G}$, are multinomial. Each point p selects exactly one stick s (enforced mathematically by the constraint $\sum_s r_{p,s} = 1$) with prior probability c_s, and each endpoint i selects one vertex j (similarly $\sum_j g_{i,j} = 1$) with probability d_j:

$$\mathrm{P}(\mathbf{R}) = \prod_{p,s} (c_s)^{r_{p,s}} \qquad \mathrm{P}(\mathbf{G}) = \prod_{i,j} (d_j)^{g_{i,j}}.$$

4.4 Learning

Given a set of observed feature point trajectories, we propose to fit this model in an entirely unsupervised fashion, by maximum likelihood learning. Conceptually, we divide learning into two challenges: recovering the skeletal structure of the model, and given a structure, fitting the model's remaining parameters. Structure learning involves grouping the observed trajectories into a number of rigid sticks, including determining the number of sticks, as well as determining the connectivity between them. Parameter learning involves determining the local geometries and motions of each stick, as well as imputing the locations of the stick endpoints and joints — all while respecting the connectivity constraints imposed by the structure.

Both learning tasks seek to optimize the same objective function—the expected complete log-likelihood of the data given the model—using different, albeit related, approaches. Given a structure, parameters are learned using the standard variational expectation-maximization algorithm. Structure learning is formulated as an "outer loop"

of learning: beginning with a fully disjoint multibody SFM solution, we incrementally merge stick endpoints, at each step greedily choosing the merge that maximizes the objective. Finally the expected complete log-likelihood can be used for model comparison and selection.

A summary of the proposed learning algorithm is provided in Figure 4.4.2.

4.4.1 Learning the model parameters

Given a particular model structure, indicated by a specific setting of $\mathbf{R}$ and $\mathbf{G}$, the remaining model parameters are fit using the variational expectation-maximization (EM) algorithm (Neal and Hinton, 1998; Dempster et al., 1977). This well-known algorithm takes an iterative approach to learning: beginning with an initial setting of the parameters, each parameter is updated in turn, by choosing the value that maximizes the expected complete log-likelihood objective function, given the values (or expectations) of the other parameters.

The objective function—also known as the negative *Free Energy* (Neal and Hinton, 1998)—is formed by assuming a fully factorized *variational posterior* distribution $\mathbb{Q}$ over a subset of the model parameters, then computing the expectation of the model's log probability (4.2) with respect to $\mathbb{Q}$, plus an entropy term:

$$\mathcal{L} = \mathrm{E}_{\mathbb{Q}}[\log \mathbb{P}] - \mathrm{E}_{\mathbb{Q}}[\log \mathbb{Q}]. \tag{4.7}$$

For this model, we define $\mathbb{Q}$ over the variables $\mathbf{V}$, $\mathbf{E}$, and $\boldsymbol{\phi}$, involved in the world-coordinate locations of the joints. The variational posterior for $\mathbf{v}_j^f$ is a multivariate Gaussian with mean parameter $\mu(\mathbf{v}_j^f)$ and precision parameter $\tau(\mathbf{v}_j^f)$, for $\mathbf{e}_i^f$ is also a Gaussian with mean $\mu(\mathbf{e}_i^f)$ and precision $\tau(\mathbf{e}_i^f)$, and for $\boldsymbol{\phi}$ is a Gamma distribution with

parameters $\alpha(\phi_j)$ and $\beta(\phi_j)$:

$$\begin{aligned}
\mathbb{Q} &= Q(\mathbf{V})\,Q(\mathbf{E})\,Q(\phi) \\
Q(\mathbf{V}) &= \prod_{f,j} \mathcal{N}(\mathbf{v}_j^f | \mu(\mathbf{v}_j^f), \tau(\mathbf{v}_j^f)^{-1}) \\
Q(\mathbf{E}) &= \prod_{f,i} \mathcal{N}(\mathbf{e}_i^f | \mu(\mathbf{e}_i^f), \tau(\mathbf{e}_i^f)^{-1}) \\
Q(\boldsymbol{\phi}) &= \prod_{j} \text{Gamma}(\phi_j | \alpha(\phi_j), \beta(\phi_j)).
\end{aligned}$$

The EM update equations are obtained by differentiating the objective function $\mathcal{L}$, with respect to each parameter, and solving for the maximum given the other parameters. We now present the parameter updates, with detailed derivation of $\mathcal{L}$ and the updates appearing in Appendix C. As a reminder, the constants appearing these equations denote: D_o the dimensionality of the observations, generally 2 but 3 will also work; F the number of observation frames; J the number of vertices; P the number of data points; S the number of sticks.

$$\begin{aligned}
\tau_w^{-1} &= \frac{\sum_{f,p,s} r_{p,s} \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2}{FPD_o} \\
\tau_m^{-1} &= \frac{\sum_{f,i} \|\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f \mathbf{k}_i\|^2}{2FSD_o} + \frac{\sum_{f,i} \tau(\mathbf{e}_i^f)^{-1}}{2FS} \\
\tau_t^{-1} &= \frac{\sum_{f=2}^{F} \sum_j \|\mu(\mathbf{v}_j^f) - \mu(\mathbf{v}_j^{f-1})\|^2}{(F-1)JD_o} + \frac{\sum_{f,j} \tau(\mathbf{v}_j^f)^{-1}}{(F-1)J} 2^{h(f)}
\end{aligned}$$

$$\begin{aligned}
\mu(\mathbf{e}_i^f) &= \frac{\tau_m \mathbf{M}_{s(i)}^f \mathbf{k}_i + \sum_j g_{i,j} \frac{\alpha(\phi_j)}{\beta(\phi_j)} \mu(\mathbf{v}_j^f)}{\tau_m + \sum_j g_{i,j} \frac{\alpha(\phi_j)}{\beta(\phi_j)}} \\
\tau(\mathbf{e}_i^f) &= \sum_j g_{i,j} \frac{\alpha(\phi_j)}{\beta(\phi_j)} + \tau_m \\
\mu(\mathbf{v}_j^f) &= \frac{\frac{\alpha(\phi_j)}{\beta(\phi_j)} \sum_i g_{i,j} \mu(\mathbf{e}_i^f) + [f > 1]\tau_t \mu(\mathbf{v}_j^{f-1}) + [f < F]\tau_t \mu(\mathbf{v}_j^{f+1})}{\frac{\alpha(\phi_j)}{\beta(\phi_j)} \sum_i g_{i,j} + \tau_t 2^{h(f)}} \\
\tau(\mathbf{v}_j^f) &= \frac{\alpha(\phi_j)}{\beta(\phi_j)} \sum_i g_{i,j} + \tau_t 2^{h(f)}
\end{aligned}$$

$$\alpha(\phi_j) = \alpha_j + \frac{FD_o}{2}\sum_i g_{i,j}$$

$$\beta(\phi_j) = \beta_j + \frac{1}{2}\sum_{f,i} g_{i,j}\|\mu(\mathbf{e}_i^f) - \mu(\mathbf{v}_j^f)\|^2 + \frac{D_o}{2}\sum_{f,i} g_{i,j}[(\tau(\mathbf{e}_i^f))^{-1} + (\tau(\mathbf{v}_j^f))^{-1}]$$

$$\alpha_j = \alpha(\phi_j)$$

$$\beta_j = \beta(\phi_j)$$

The update for the motion matrices is slightly more challenging due to the orthogonality constraint on the rotations. A straightforward approach is to separate the rotation and translation components of the motion and to solve for each individually. The update for translation is obtained simply via differentiation:

$$\mathbf{M}_s^f = \begin{bmatrix}\mathbf{R}_s^f & \mathbf{t}_s^f\end{bmatrix}$$

$$\mathbf{t}_{s,f} = \Big(\tau_w \sum_p r_{p,s}(\mathbf{w}_p^f - \mathbf{R}_s^f \mathbf{l}_{s,p}) + \tau_m \sum_{\{i|s(i)=s\}} (\mu(\mathbf{e}_i^f) - \mathbf{M}_s^f \mathbf{k}_{s,i})\Big) / \Big(\tau_w \sum_p r_{p,s} + 2\tau_m\Big)$$

To deal with the orthogonality constraint on $\mathbf{R}_s^f$, its update can be posed as an *orthogonal Procrustes problem* (Golub and Van Loan, 1996; Viklands, 2006). Given matrices $\mathbf{A}$ and $\mathbf{B}$, the goal of orthogonal Procrustes is to obtain the matrix $\mathbf{R}$ that minimizes $\|\mathbf{A}-\mathbf{R}\mathbf{B}\|^2$, subject to the constraint that the rows of $\mathbf{R}$ form an orthonormal basis. Computing the most likely rotation involves maximizing the likelihood of the observations (4.3) and of the endpoints (4.4), which can be written as the minimization of $\sum_p \|(\mathbf{w}_p^f - \mathbf{t}_{s,f}) - \mathbf{R}_s^f \mathbf{l}_{s,p}\|^2$ and $\sum_{\{i|s(i)=s\}} \|(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s,f}) - \mathbf{R}_s^f \mathbf{k}_{s,i}\|^2$ respectively. Concatenating the two problems together, weighted by their respective precisions, allows the update of $\mathbf{R}_s^f$ to be written as a single orthogonal Procrustes problem $\operatorname{argmin}_{\mathbf{R}_s^f} \|\mathbf{A} - \mathbf{R}_s^f \mathbf{B}\|^2$, where

$$\mathbf{A} = \begin{bmatrix}\left[\sqrt{\tau_w}\, r_{p,s}(\mathbf{w}_p^f - \mathbf{t}_{s,f})\right]_{p=1..P} & \left[\sqrt{\tau_m}\,(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s,f})\right]_{\{i|s(i)=s\}}\end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix}\left[\sqrt{\tau_w}\, r_{p,s}\mathbf{l}_{s,p}\right]_{p=1..P} & \left[\sqrt{\tau_m}\,\mathbf{k}_i\right]_{\{i|s(i)=s\}}\end{bmatrix}.$$

The solution is to compute the singular value decomposition of $\mathbf{B}\mathbf{A}^\top \overset{SVD}{=} \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, and let $\mathbf{R} = \mathbf{V}\,\mathbf{I}_{m\times n}\mathbf{U}^\top$, where m and n are the numbers of rows in $\mathbf{A}$ and $\mathbf{B}$ respectively.

Given $\mathbf{R}_s^f$ and $\mathbf{t}_s^f$, the updates for the local coordinates are:

$$\mathbf{l}_{s,p} = \big(\sum_f \mathbf{R}_s^{f\top}\mathbf{R}_s^f + \frac{\tau_p}{\tau_w}\mathbf{I}\big)^{-1} \sum_f \mathbf{R}_s^{f\top}(\mathbf{w}_p^f - \mathbf{t}_{s,f})$$

$$\mathbf{k}_i = \big(\sum_f \mathbf{R}_{s(i)}^{f\top}\mathbf{R}_{s(i)}^f + \frac{\tau_p}{\tau_m}\mathbf{I}\big)^{-1} \sum_f \mathbf{R}_{s(i)}^{f\top}(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s(i)}^f)$$

The final issue to address for EM learning is initialization. Many ways to initialize the parameters are possible; here we settle on one simple method that produces satisfactory results. The motions and local coordinates, $\mathbf{M}$ and $\mathbf{L}$, are initialized by solving SFM independently for each stick (Tomasi and Kanade, 1992). The vertex locations are initialized by averaging the observations of all sticks participating in the joint: $\mu(\mathbf{v}_j^f) = (\sum_{i,p} g_{i,j}\, r_{p,s(i)}\, \mathbf{w}_p^f)/(\sum_{i,p} g_{i,j}\, r_{p,s(i)})$. The endpoints are initially coincident with their corresponding vertices, $\mu(\mathbf{e}_i^f) = \sum_j g_{i,j}\, \mu(\mathbf{v}_j^f)$, and the $\mathbf{K}$s by averaging the back-projected endpoint locations: $\mathbf{k}_i = \frac{1}{F}\sum_f \mathbf{R}_{s(i)}^{f\top}(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s(i)}^f)$. All precision parameters are initialized to constant values, as discussed in Section 4.5.1.

4.4.2 Learning the skeletal structure

Structure learning in this model entails estimating the assignments of feature points to sticks (including the number of sticks), and the connectivity of sticks, expressed via the assignments of stick endpoints to vertices. The space of possible structures is enormous. We therefore adopt an incremental approach to structure learning: beginning with a fully disconnected multibody-SFM model, we greedily add joints between sticks by merging vertices. After each merge the model parameters are updated via EM, and the assignments of observations to sticks are resampled. After performing the desired number of merges, model selection—that is, choosing the optimal number of joints—is guided by comparing the expected complete log-likelihood of each model.

The first step in structure learning involves hypothesizing an assignment of each observed feature trajectories to a stick. This is accomplished by clustering the trajectories using the *Affinity Propagation* algorithm of Frey and Dueck (2007). Affinity propagation

takes as input an affinity matrix, for which we supply the affinity measure from Yan and Pollefeys (2006a,b) as presented in Section 4.2.1 (or for 3D data, Kirk et al. (2005) discussed in 4.2.2). During EM parameter learning, the stick assignments $\mathbf{R}$ are resampled every 10 iterations using the posterior probability distribution

$$\mathrm{P}(r_{p,s}) \propto c_s \exp\left(-\frac{\alpha_w}{2}\sum_f \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2\right) \quad \text{s.t.} \sum_{s'} r_{p,s'} = 1.$$

Instead of relying only on information available before model fitting begins (Costeira and Kanade, 1998; Kirk et al., 2005; Yan and Pollefeys, 2006b), resampling of stick assignments allows model probability to be improved by leveraging current best estimates of the model parameters.

The second step of structure learning involves determining which sticks endpoints are joined together. As discussed earlier, connectivity is captured by assigning stick endpoints to vertices; each endpoint must be associated to one vertex, and vertices with two or more endpoints act as articulated joints. (Valid configurations include only cases in which endpoints of a given stick are assigned to different vertices.) We employ an incremental greedy scheme for inferring this graphical structure $\mathbf{G}$, beginning from an initial structure that contains no joints between sticks. Thus, in terms of the model, we start with $J = 2S$ vertices, one per stick-endpoint, so $g_{i,j} = 1$ if and only if $j = i$. Given this initial structure, parameters are fit using variational EM.

A joint between sticks is introduced by merging together a pair of vertices. The choice of vertices to merge is guided by our objective function $\mathcal{L}$. At each stage of merging we consider all valid pairs of vertices, putatively joining them and estimating (via 20 iterations of EM) the change in log-likelihood if this merge were accepted. The merge with the highest log-likelihood is performed, by modifying $\mathbf{G}$ accordingly, and the model parameters are re-optimized with 200 additional iterations of EM, including resampling of the stick assignments $\mathbf{R}$. This process is repeated until no valid merges remain, or the desired maximum number of merges has been reached.

Computational Cost

By examining the updates presented in Section 4.4.1, in can be seen that the cost of each iteration of EM parameter learning scales linearly in the following quantities: F the number of frames, J the number of joints, P the number of observed feature point trajectories, and S the number of sticks. (Note that since the number of rows in $\mathbf{A}$ and $\mathbf{B}$ are fixed, each orthogonal Procrustes update of $\mathbf{R}_s^f$ has a cost that is linear in P—the initial multiplication $\mathbf{AB}^\top$—in addition to a constant-cost SVD and final multiplication.)

Each stage of greedy merging requires computing the expected log-likelihood for all of the possible pairs of vertices to be merged. The number of possible merges scales with $O(J^2)$, which, since $J = 2S$ during the first stage, can be as high as $4S^2$. In practice, however, it is possible to reduce the number that must be considered. Savings can be obtained by noting the symmetry of the merge operation, reducing the number of unique merges by a factor of two, as well as by disallowing self-merges between the two endpoints of a stick. A less obvious savings can be realized by avoiding duplication when merging with a stick that has two free endpoints, since the change in probability from merging to either of these otherwise unconstrained endpoints will be identical. During the initial stage, when the structure contains no joints, this reduces the number of unique merges by an additional factor of four. During later stages, there are fewer possible merges to consider since J, the number of vertices, decreases by one for each stage, and our previously mentioned restriction—that the endpoints of a stick cannot be assigned to the same vertex—eliminates a greater proportion of potential merges.

In our experiments these optimizations are sufficient to yield acceptable runtimes, however given much larger models the number of possible merges could be reduced to $O(J)$ by allowing each stick to merge with only a fixed number (*e.g.* five) of its nearest neighbors. It may also be possible to achieve further savings through caching—approximating the expected change in log-likelihood of a merge with its value from the previous stage, without recomputing (Ross et al., 2007).

1. Obtain an initial grouping $\mathbf{R}$ by clustering the observed trajectories using Affinity propagation. Initialize $\mathbf{G}$ to a fully disconnected structure.

2. Optimize the parameters $\mathbf{M}$, $\mathbf{L}$, $\mathbf{K}$, $\mathbf{V}$, $\boldsymbol{\phi}$, $\mathbf{E}$, using 200 iterations of the variational EM updates, resampling $\mathbf{R}$ every 10 iterations.

3. For all vertex-pair merges, estimate gain resulting from the proposed structure by updating the parameters with 20 EM iterations and noting the change in expected log-probability.

4. Choose the merge with the largest gain, modifying $\mathbf{G}$ accordingly. Re-optimize parameters using another 200 EM iterations, resampling $\mathbf{R}$ every 10^{th}.

5. Go to Step 3 and repeat. Exit when there are no more valid merges, or the maximum number of merges has been reached.

Figure 4.5: A summary of the learning algorithm

4.5 Experimental Results and Analysis

We now present results of the proposed algorithm on a range of different feature point trajectory datasets. This includes data obtained by automatically tracking features in video, from optical motion capture (both 2D and 3D), as well as a challenging artificially generated sequence. In each experiment a model was learned on the first 70% of the sequence frames, with the remaining 30% held out as a test set used to measure the model's performance. Learning was performed using the algorithm summarized in Figure 4.4.2, with greedy merging continuing (generally) until no valid merges remained. After each stage of merging, we saved the learned model and corresponding expected complete log-likelihood—the objective function learning maximizes. The likelihoods were plotted for comparison, and used to select the optimal model.

The learned model's performance was evaluated based on its ability to impute (reconstruct) the locations of missing observations. For each test sequence we generated a set of missing observations by simulating an occluder that sweeps across the scene, obscuring points as it passes. We augmented this set with an additional 5% of the observations chosen to be "missing at random", to simulate drop-outs and measurement errors, resulting in a overall occlusion rate of 10-15%. The learned model was fit to the un-occluded points of the test sequence, and used to predict the location of the missing points. Performance was measured by computing the root-mean-squared error between the predictions and the locations of the heldout points. We compared the performance of our model against similar prediction errors made by single-body and multibody structure from motion models.

This section begins with a brief analysis of the effect of precision parameters during learning, followed by experimental results on five datasets: a video of an excavator, a video of a walking giraffe, 2D feature trajectories obtained from human motion capture, an synthetic dataset of a jointed ring, and an additional set of human motion data in 3D. Finally we conclude with a brief comparison against two related methods (Yan and

Pollefeys, 2006b; Kirk et al., 2005).

4.5.1 Setting Precision Parameters During Learning

As presented, the model contains a number of precision parameters to be determined during learning: τ_w, τ_m, τ_t, τ_p, $\tau(\mathbf{v}_j^f)$, $\tau(\mathbf{e}_i^f)$, as well as the parameters of the prior distribution on the joint prior, α_j and β_j. In practice, simply initializing these precisions to arbitrary values and allowing them to adapt freely during EM leads to poor results. Some of the precisions—particularly α_w, α_m, $\tau(\mathbf{v}_j^f)$, and $\tau(\mathbf{e}_i^f)$—tend to grow unbounded, thus we have found it useful to specify a maximum precision of 50 (a standard trick during EM). In contrast, the joint precisions ϕ_j (given by $\alpha(\phi_j)/\beta(\phi_j)$) tend towards relatively small values, resulting in a model that has very little cohesion in the joints. To counteract this we specify a very strong prior on ϕ_j encouraging it towards large values: $\alpha_j = 2 \times 10^5 \times$ *maximum precision* and $\beta_j = 10^5$, resulting in an expected value of $2 \times$ *maximum precision* with limited variance. When fitting the motion of a stick, assuming other precisions saturate at the maximum, this means that keeping an endpoint near its vertex is at least twice as important as keeping a feature point near its observed location.

In our experiments, we have found temporal smoothing of the vertices, governed by precision τ_t to be a disadvantage during learning. Particularly at the beginning, when the structure contains no joints, smoothing causes the unconnected vertices and endpoints to drift away from the actual observations at each frame, towards their temporal mean. Thus, in all of the following experiments we disable smoothing during learning. However, when measuring test performance it's not uncommon for one or more adjacent sticks to be entirely occluded during a frame. When this happens, smoothed locations of the vertices provide the only source of information about the location of the stick, and thus temporal smoothing is essential for limiting test error. When measuring test performance, therefore, we enable smoothing and set $\tau_t = 2000$.

Finally, the precisions play an important role in determining the optimal number of joints. During model selection we seek the model with the largest expected complete log-likelihood, hoping that this will include as many plausible joints as possible. However most terms in this objective function favour a disconnected model, with more vertices and fewer joints. To understand this, consider the problem of estimating the motion of an unconnected stick. Since there are no constraints on the vertices, they can be trivially placed to be coincident with endpoints, thus the motion variable needs only focus on maximizing the probability of the observations. However when two sticks are joined together, perfect placement of the vertices is generally not possible, requiring modelling compromises that introduce slight reductions in observation probability. The one term in the objective function that does not decrease as merges are performed is the entropy of the vertices $\mathrm{E}_{Q(\mathbf{V})}[\log Q(\mathbf{V})]$.

Assuming each precision parameter in $Q(v)$ is equal to the maximum precision, p, this entropy is $(FJD_o/2)\log(2\pi e/p)$. If p is greater than $2\pi e \approx 17.08$, then the differential entropy is negative[5]. The result is that decreasing the number of vertices J causes the log-likelihood to increase. In effect a fixed cost of $(FD_o/2)\log(2\pi e/p)$ is paid for each vertex in the model, giving us the desired bias towards connectivity. Plots of the relevant log-likelihood terms are included for the datasets presented below.

When applying the approach to new datasets, in practice the quality of the learned solution can sometimes be dependent of the value of the *maximum precision* parameter. Using the above analysis as a guide, a parameter search over a coarse range of values from approximately 18 to 50 can be helpful for obtaining the best possible results.

[5]Although unintuitive, negative differential entropies are perfectly acceptable (Cover and Thomas, 1991).

4.5.2 Excavator

Our first dataset consisted of a video clip of an excavator. We used a Kanade-Lucas-Tomasi tracker (Shi and Tomasi, 1994) with manual clean-up to obtain 35 feature trajectories across 176 frames. Our algorithm processed the data in 4 minutes on a 2.8 GHz processor. The learned model at each stage of greedy merging is depicted in Figure 4.6. The optimal structure was chosen by comparing the log-likelihood at each stage, as plotted in Figure 4.7 (left). The four most significant terms comprising this objective function are plotted individually in Figure 4.7 (right). As can be seen, joining sticks adds additional constraints that reduce the expected probability of the observations (top left), the endpoints given vertices (top right), and the endpoints given $\mathbf{Mk}$ (bottom left). In contrast the vertex entropy term (bottom right) acts as a per-vertex penalty, which decreases as we merge vertices, favoring more highly connected models. Figure 4.7 (bottom) shows that the system's prediction error for occluded data was significantly better than either multibody or single-body SFM.

As can be seen in Figure 4.6, the model does a good job at recovering the structure—the grouping and connectivity—of the observed trajectories. The reconstruction shows some deviation between the inferred locations of the joints and their intuitive positions. Nevertheless, the model is fully able to capture the range of motion exhibited by the excavator's arm.

Using the excavator data, we also examined the model's robustness to learning with occlusions in the training data. When the occlusion scheme described earlier was employed to generate a training set with missing observations, and the learning algorithm was applied to this data, it was still able to recover the correct structure. Similarly, when training observations were randomly withheld during training, rather than using structured occlusion, the correct structure was reliably recovered with up to 75% of the training observations missing.

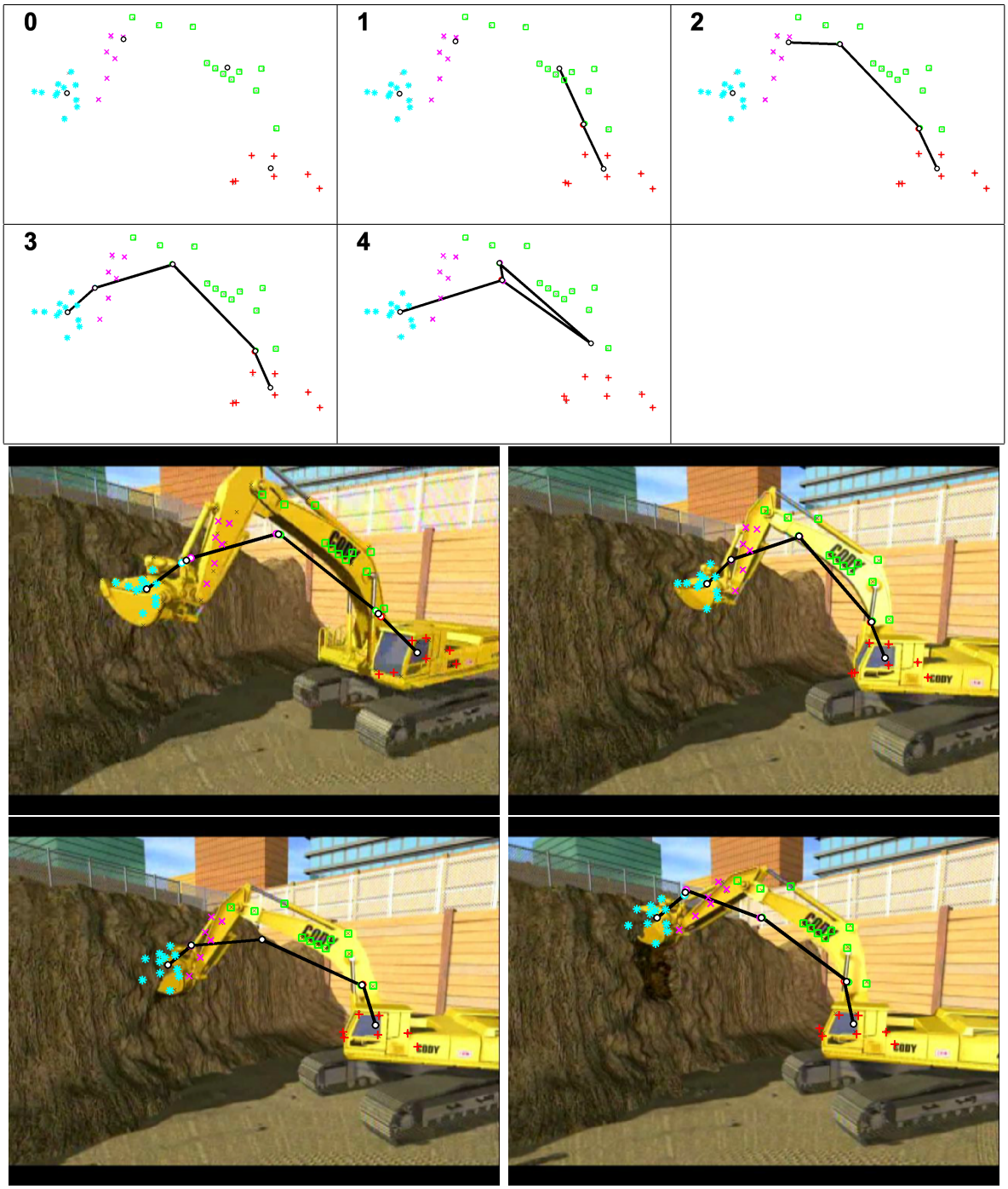

Figure 4.6: Excavator Data: Shown at the top are the models learned in each of the five successive stages of greedy learning. Reconstructions of the observed markers are shown with different symbols depending on their stick assignments. The locations of vertices are shown as black o's, and black lines are drawn to connect each stick's pair of vertices. At the bottom, the selected model (stage 3) is used to reconstruct the observed feature trajectories, and the results are superimposed over the corresponding frames of the input video.

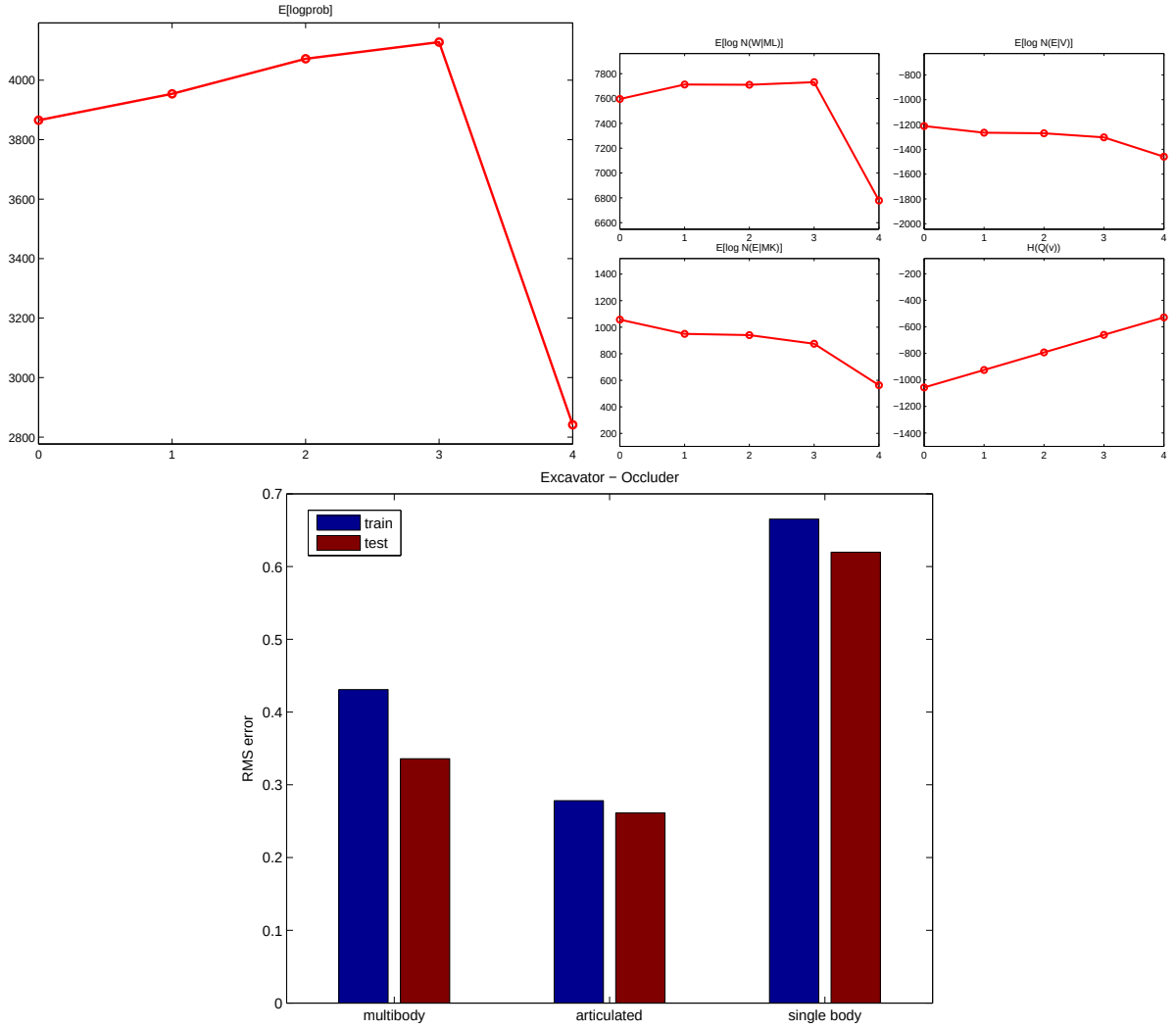

Figure 4.7: Excavator Log-likelihood and Error. At the top-left we see that stage 3 of merging produces the model with the highest log-probability. At the top-right are individual plots of the four most significant terms comprising the log probability. At the bottom, we can see that the learned model exhibits less reconstruction error than either single or multibody SFM models.

4.5.3 Giraffe

Our second dataset consisted of a video of a walking giraffe. As before features were tracked, producing 60 trajectories across 128 frames. Merging results are depicted in Figure 4.8. Using the objective function to guide model selection (Figure 4.9), the best structure corresponded to stage 10, and this model is shown superimposed over the original video in Figure 4.1, appearing at the start of this chapter.

4.5.4 2D Human

Our third dataset consisted of optical human motion capture data (courtesy of the Biomotion Lab, Queen's University, Canada), which we projected from 3D to 2D using an isometric projection. The data contained 53 features, tracked across a 1018-frame range-of-motion exercise (training data), and 318 frames of running on an inclined plane (test data). The structures learned during greedy merging are shown in Figure 4.10, of which stage 11 most closely matches human intuition.

By examining the plots in Figure 4.11, it can be noted that the expected log-likelihood of the various models forms a plateau, roughly between stages 8 and 11, rather than a sharp peak as seen for the Excavator data. Although stage 11 is not actually the most likely model (stage 8 is slightly higher), the log-likelihood decreases rapidly after stage 11. This suggests that having too many joints—and thereby hampering the ability of sticks to move so as to fit the observations—is a bigger disadvantage to the model than simply having too few joints. Theoretically it may be possible to encourage a global maximum in log-likelihood at stage 11 by simply increasing the maximum precision (thereby penalizing stage 8 which has more vertices). However, recognizing our preference for models with as many plausible joints as possible, selecting the stage at the edge of the plateau—stage 11—seems a reasonable choice.

Again, the articulated model achieved a lower test error than either SFM or multibody

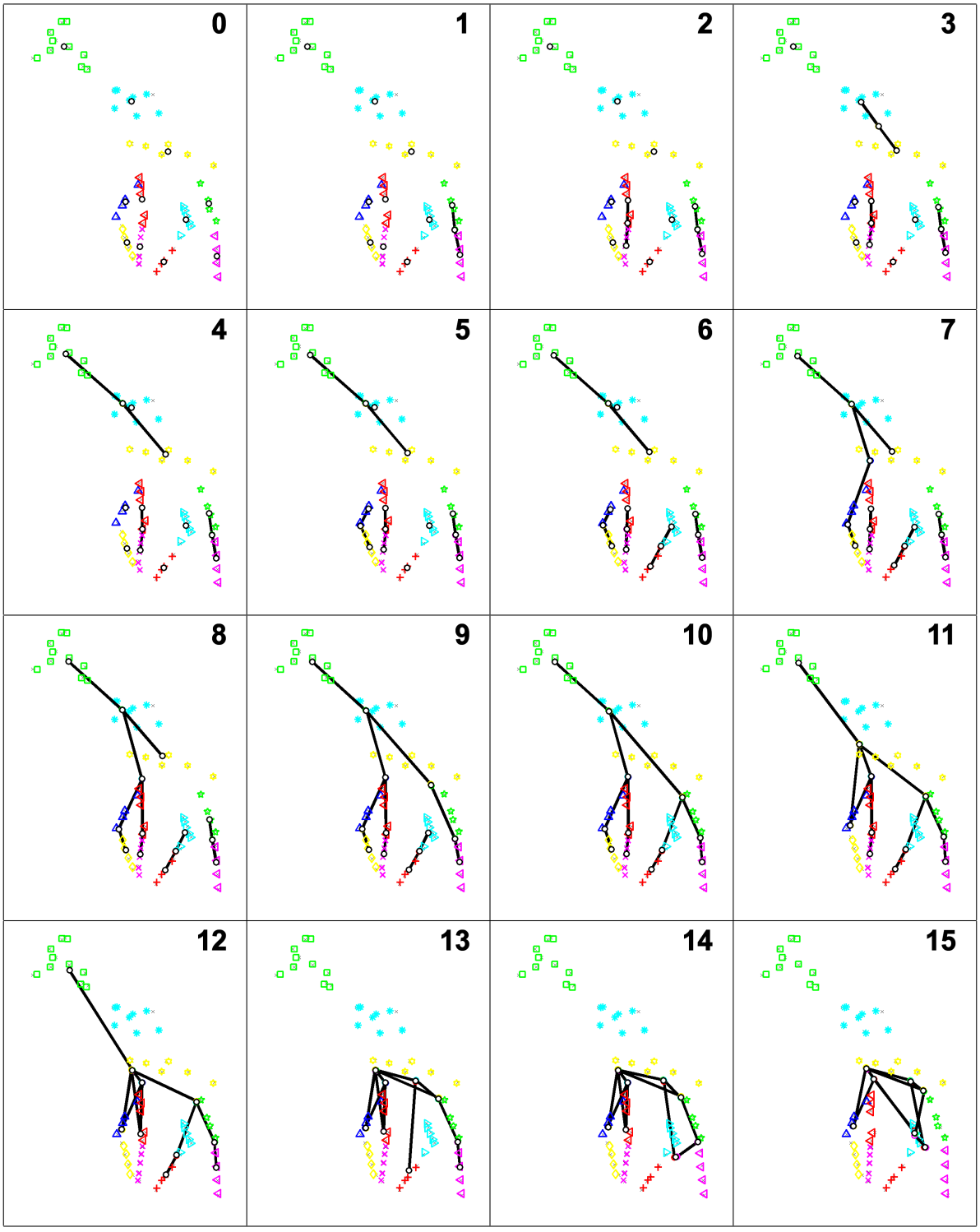

Figure 4.8: Giraffe structures learned during greedy merging. Stage 10 has the highest expected log-likelihood.

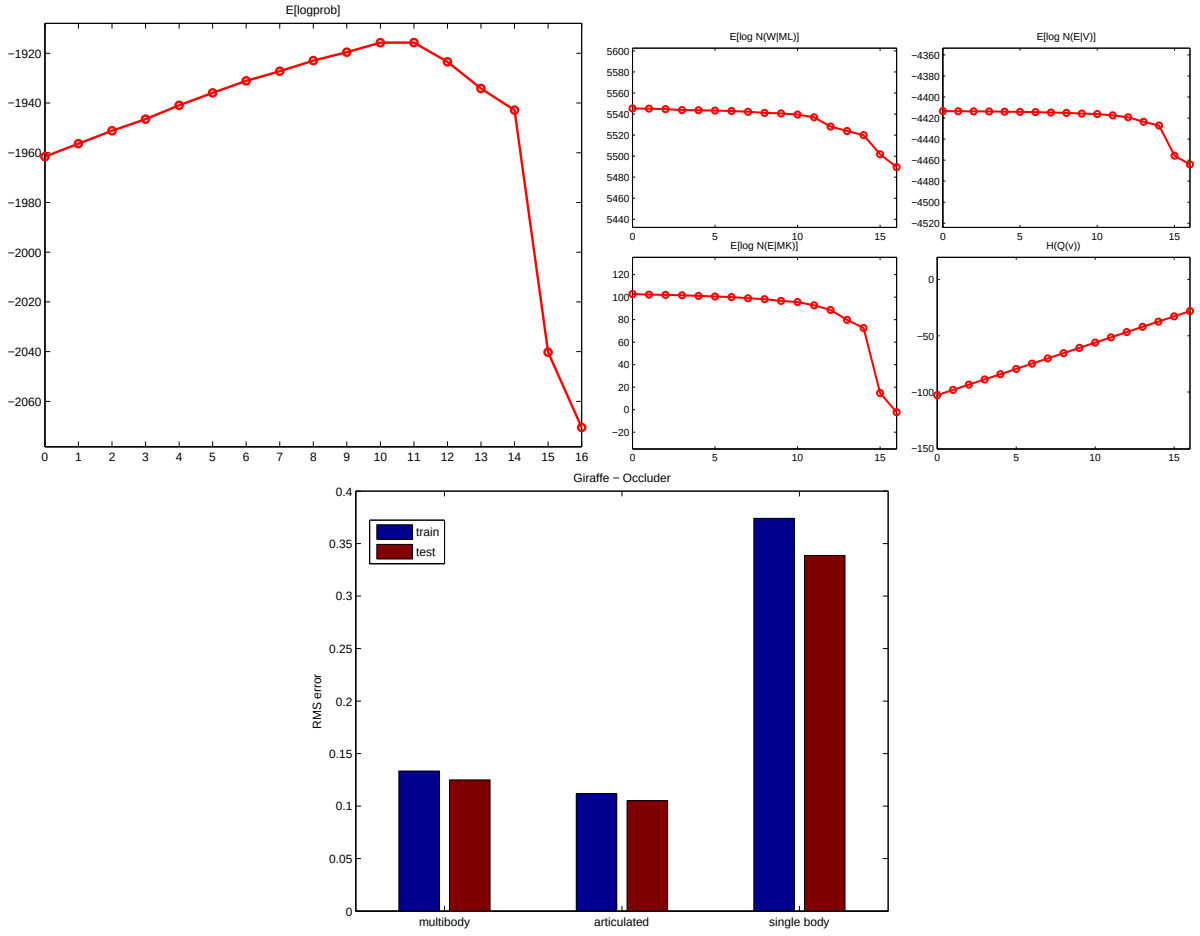

Figure 4.9: Giraffe Log-likelihood and Error.

SFM.

4.5.5 Synthetic Ring

In order to evaluate the performance of the model on data which contains significant out-of-plane motion, we created a challenging synthetic dataset which depicting a segmented ring deforming in space. The generated sequence consisted of 100 features across 300 frames, to which independent Gaussian noise of standard deviation 0.05 was added. (For comparison, each stick was approximately 0.5 units wide and 5 units long.) Six frames from the sequence are depicted in Figure 4.12.

The models learned for the successive stages of merging are shown in Figure 4.13. The sharp downturn in log-likelihood between stages 8 and 9, shown in Figure 4.14, suggests selecting stage 8 as the best model. (Note that although stage 0, which is equivalent to multibody SFM, has a higher expected log-probability, stage 8 has the lower test error.) Unlike methods based on spanning trees, our approach was able to recover the correct closed ring structure.

Interestingly, all of the learned structures chose to group the feature points into eight sticks, three more than were in the true grouping used to generate the data, as illustrated in the bottom-right of Figure 4.13. Examination of the results show that these extra groups arise from splitting three of the true sticks each into a pair sticks connected by a joint. Although the learned structure is an over-segmentation of the ground truth structure, it still provides a perfectly acceptable model of the data.

4.5.6 3D Human

Although recovering 3D structure from 3D observations is much simpler than from 2D data, it also receives attention in the literature. As mentioned previously, our model easily extends to 3D observations, so we include an additional experiment demonstrating this ability. Here we trained our model on optical human motion capture data obtained

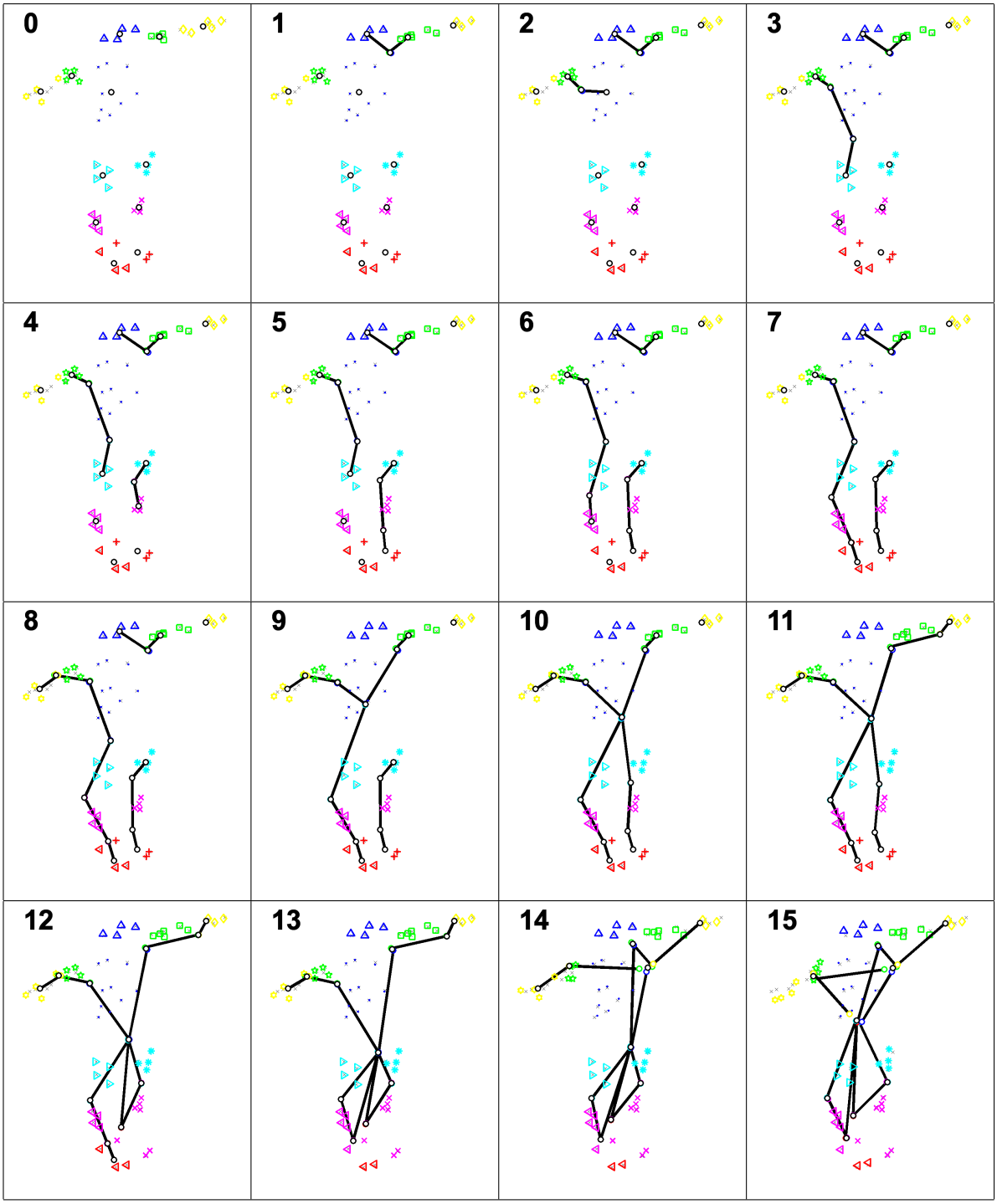

Figure 4.10: 2D Human structures learned during greedy merging, of which stage 11 most closely matches human intuition.

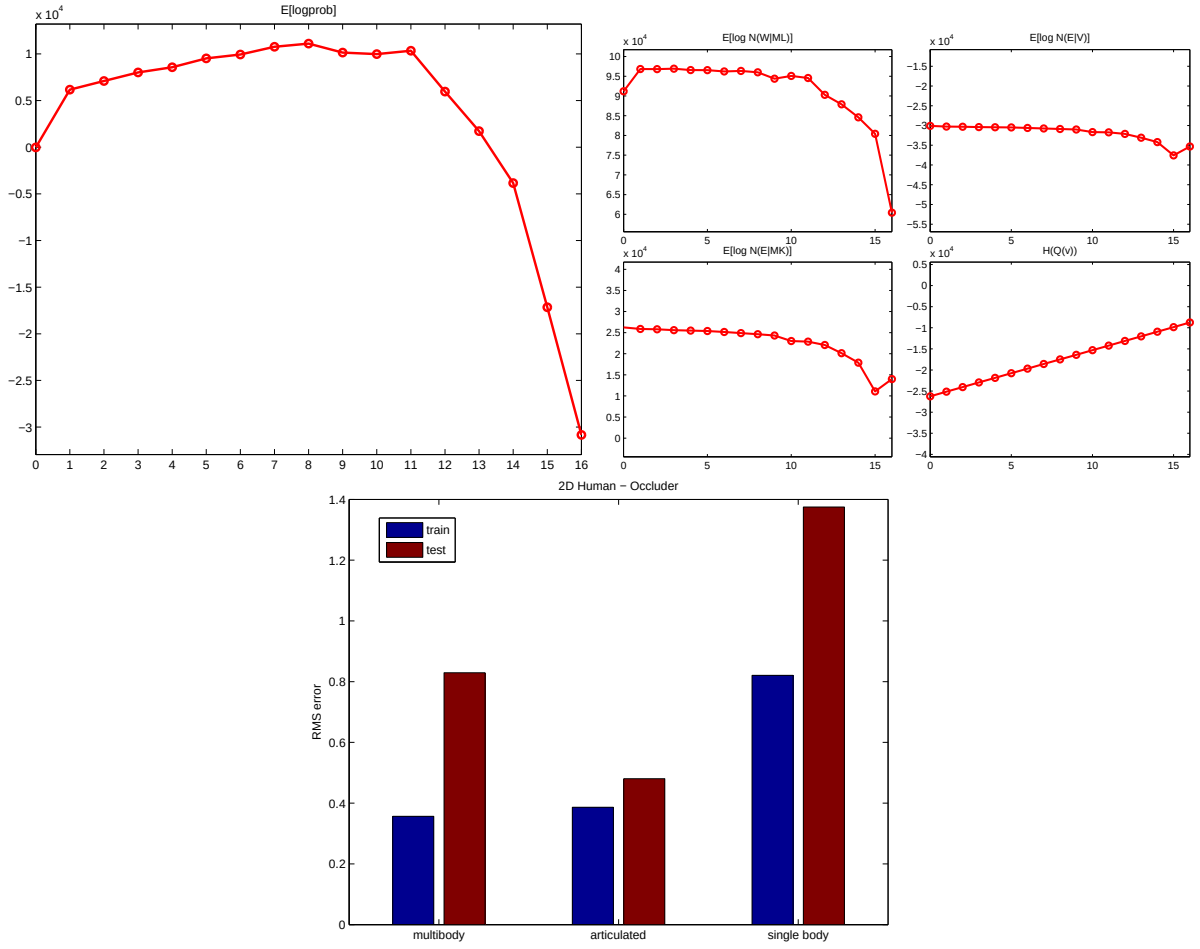

Figure 4.11: 2D Human Log-likelihood and Error.

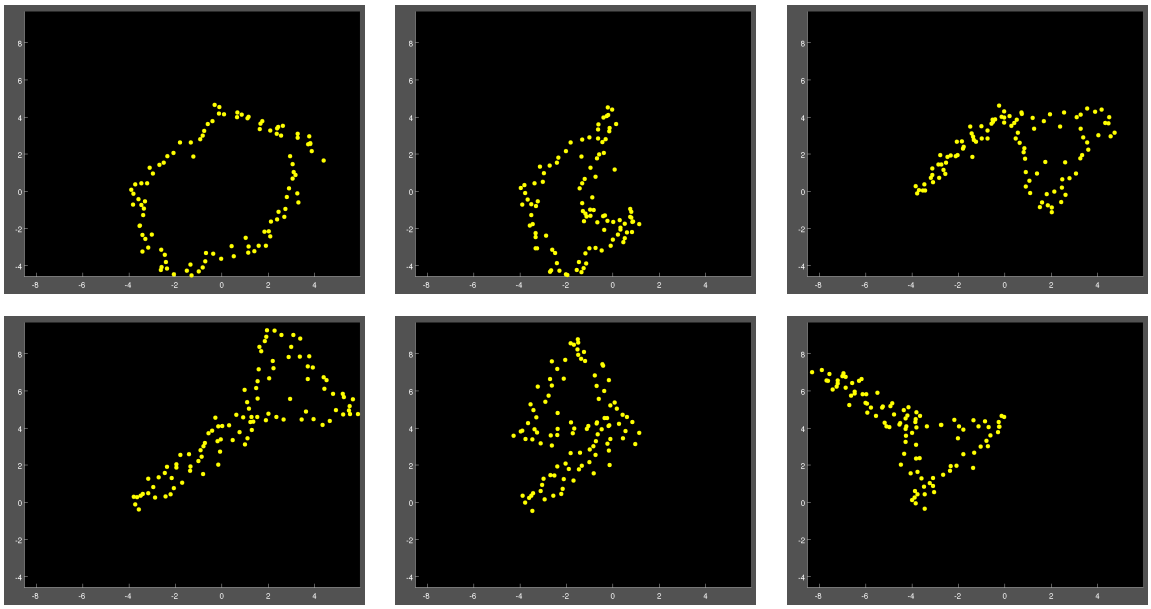

Figure 4.12: Synthetic Ring Data: Six frames selected from a synthetic data sequence depicting the motion of a 5-segmented ring. The ring undergoes significant out-of-plane motion.

from the Carnegie Mellon University Motion Capture Database. The data consisted of 174 feature points tracked across 732 frames (downsampled by a factor of three from the original framerate). The results of greedy merging are shown in Figure 4.15, and the corresponding log-likelihoods in Figure 4.16. Since learning from 3D observations is an easier problem, the most likely structure—stage 15—is visually more appealing than the structure learned earlier on the 2D human data.

4.5.7 Related Methods

Finally, as an additional qualitative comparison, we re-implemented the methods of Yan and Pollefeys (2006b) and Kirk et al. (2005), and ran them on a selection of the datasets.

The method of Yan and Pollefeys was trained on the Giraffe and 2D Human datasets, and the resulting structures are shown in Figure 4.17, at the top-left and top-right respectively. (The performance of the method is sensitive to the values of manually tuned

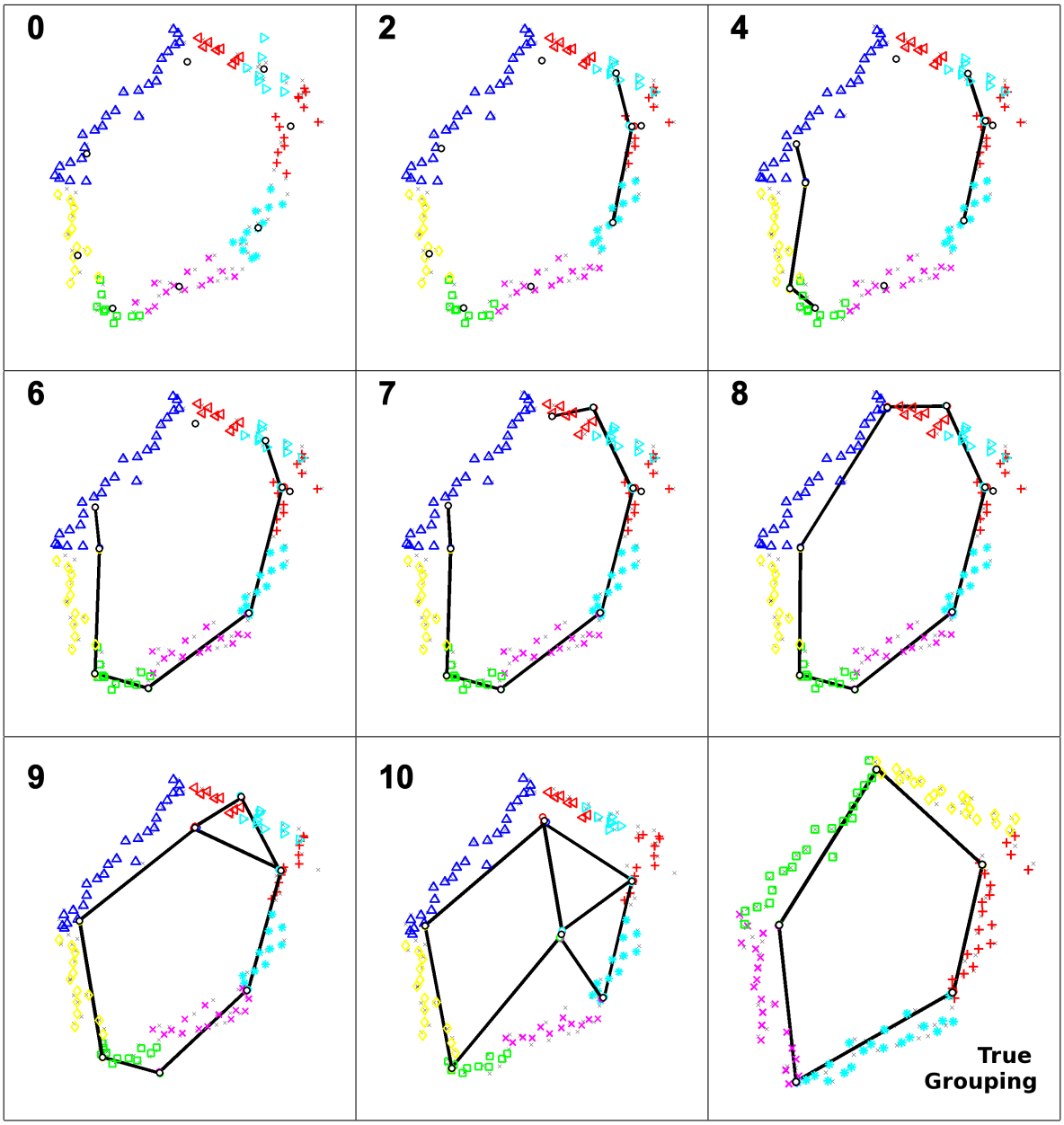

Figure 4.13: Synthetic Ring Structures learned during greedy merging, of which stage 8 is the best. In comparison to the ground-truth structure, shown in the lower-right, the learned model over-segments the data into 8 sticks, rather than 5. However, since this involves splitting three of the true sticks in half, the learned model still provides a good fit to the data.

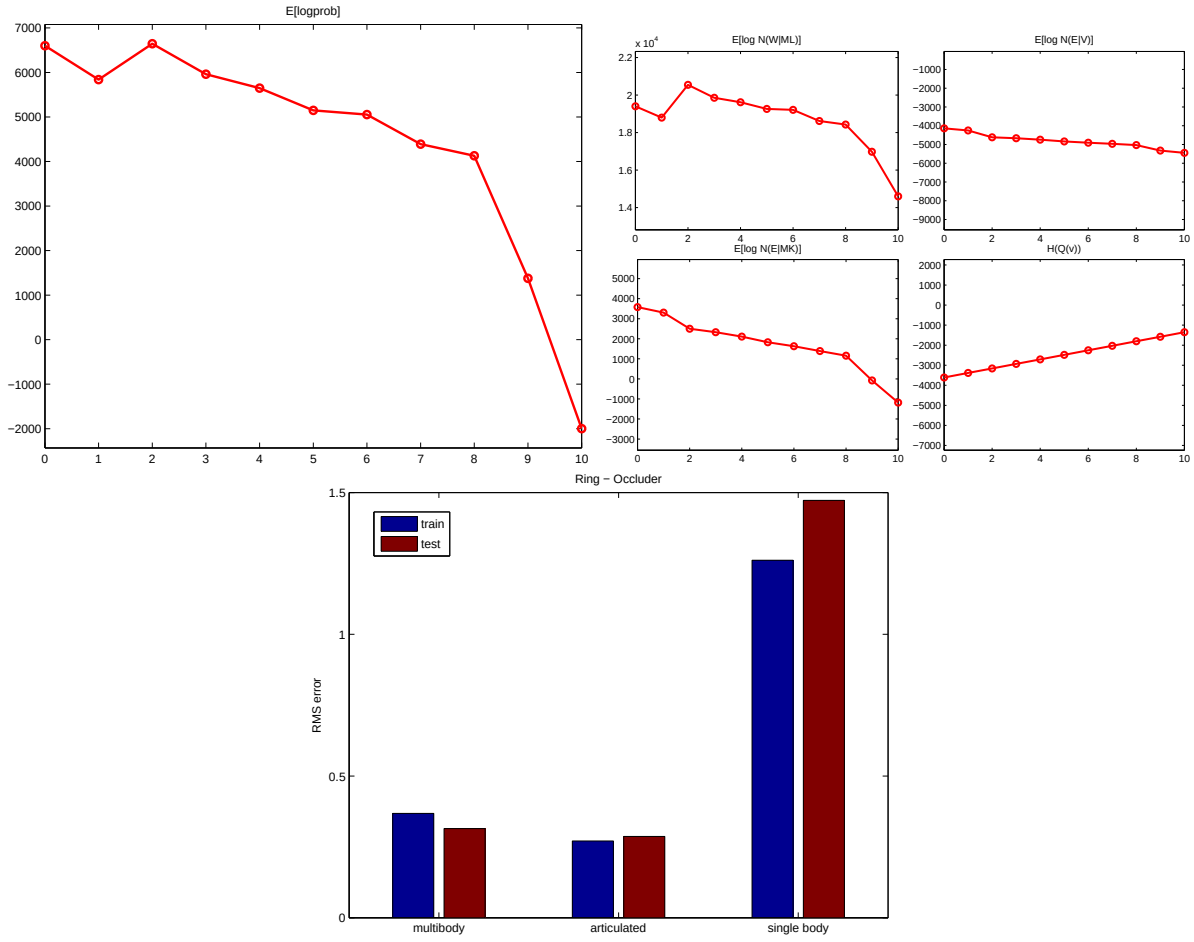

Figure 4.14: Synthetic Ring Log-likelihood and Error. The sharp downturn in log-likelihood at stage 9 suggests selecting the structure learned during stage 8.

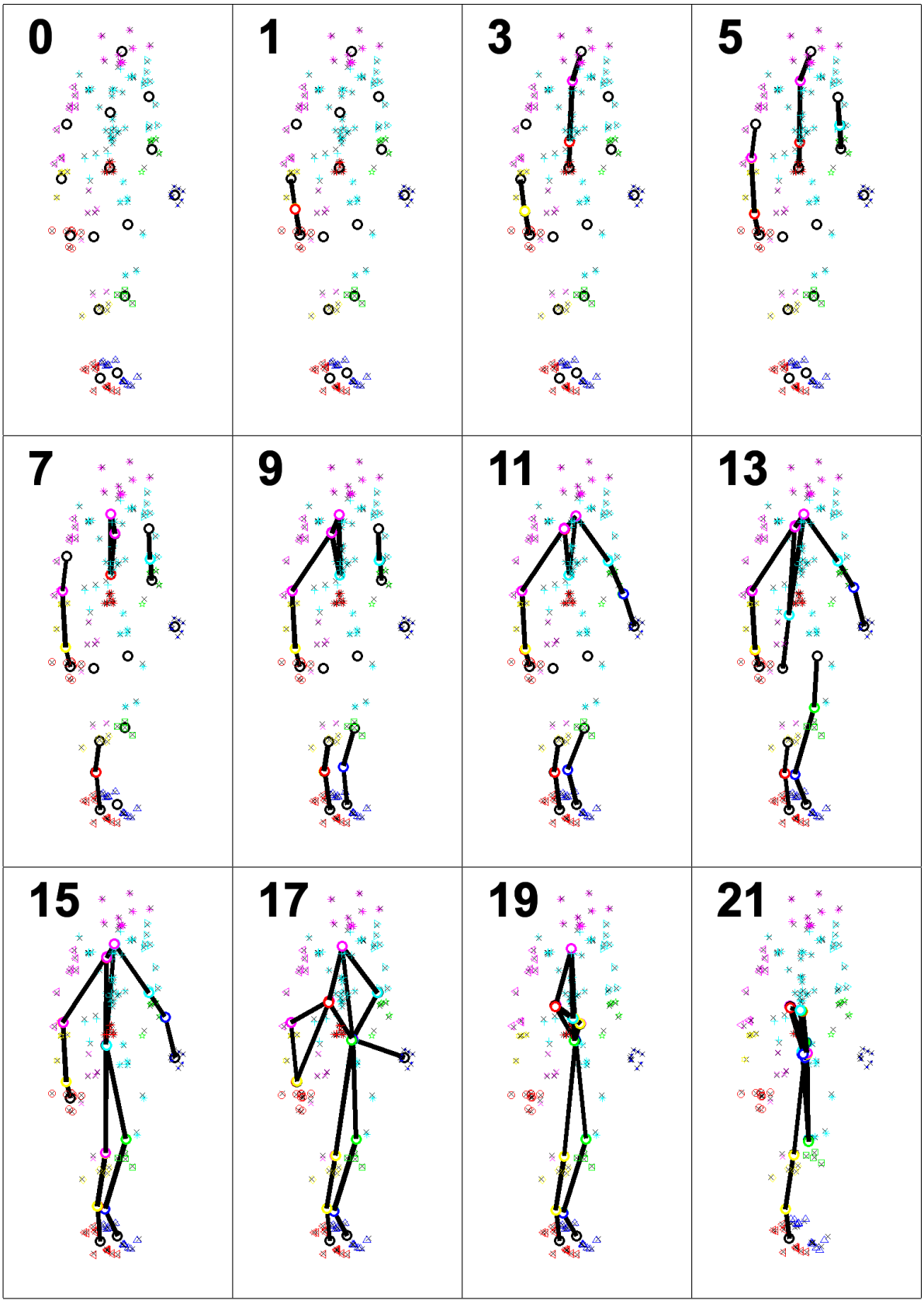

Figure 4.15: 3D Human structures learned during greedy merging. Stage 15 has the highest log-likelihood.

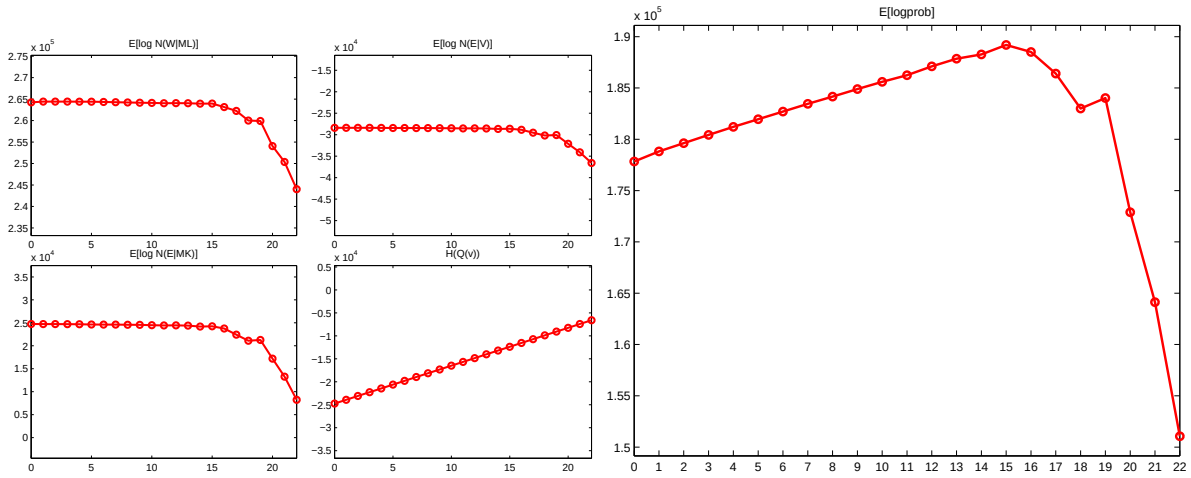

Figure 4.16: 3D Human Log-likelihood.

parameters. In each case we searched over a small range of parameter values and selected the resulting structure that was most visually appealing.) A weakness of the Yan and Pollefeys method is that it chooses to connect sticks based on the degree of linear-dependence between their motions, rather than the ability to identify a consistent joint between them. As a result, it twice connects fore- and hind-leg segments which move in phase as the giraffe walks, despite the implausibility of a joint between them.

As described earlier, the method of Kirk et al. is designed to work on 3D optical motion capture data, thus we trained it on the 3D Human dataset used in Section 4.5.6, as well as the on 3D feature locations that gave rise to the 2D Human dataset from Section 4.5.4. In the original paper, Kirk et al. focus on fitting their model to "calibration" sequences, in which the actor fully flexes each of his individual joints. Indeed, as shown in Figure 4.17, lower-right, the method does a good job at recovering the structure from the range-of-motion sequence. (For comparison, the results of our method trained on the 2D-projection of the same sequence is shown in Figure 4.10.) In contrast, on the other 3D Human sequence which depicts walking and sitting rather than range-of-motion exercises, Kirk's method fares much more poorly (Figure 4.17 lower-left, *c.f.* our method Figure 4.15).

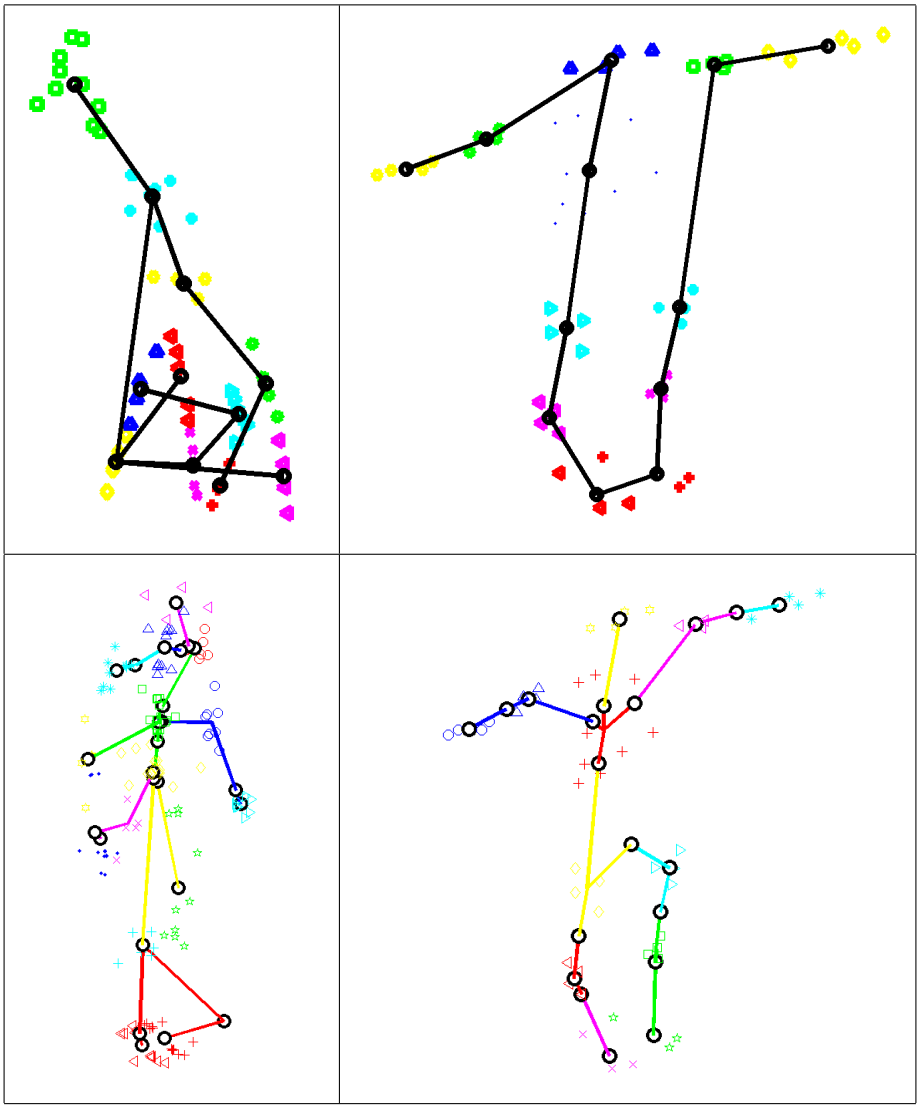

Figure 4.17: A comparison of results by related methods. The Yan and Pollefeys (2006b) method was trained on 2D feature locations from the giraffe video and the 2D Human motion capture sets (top row), while the Kirk et al. (2005) method was trained on 3D feature locations from the two datasets of human motion capture (bottom row) .

4.6 Discussion

We have demonstrated a single coherent model that can learn the structures and motion of articulated skeletons. This model can be applied to a variety of structures, requiring no input beyond the observed feature trajectories, and a minimum of manually adjusted precision parameters.

Our model makes a number of contributions to the state of the art. First, it is based optimizing a single global objective function, which details how all aspects of learning—grouping, connectivity, and parameter fitting—contribute to the overall quality of the model. Having this objective function permits iteration between updates of the structure and parameters, allowing information obtained from one stage to assist learning in the other. Moreover, the value of the objective function proves useful for model selection, determining the optimal number of joints. Also, the noise in our generative model plays an important role, allowing a degree of non-rigidity in the motion with respect to the learned skeleton. This not only allows a feature point to move in relation to its associated stick, but also permits complexity in the joints, as the stick endpoints joined at a vertex need not coincide exactly. In addition we presented a method for quantitative comparison, based on imputing the locations of occluded observations, and were able to demonstrate that our model performs measurably better than single-body or multibody structure from motion.

To obtain good results, the model requires a certain density of features, in particular because the affinity matrix used for initialization (Yan and Pollefeys, 2006a) requires at least 4 points per stick. In addition, the flexibility of learned models is limited to the degrees of freedom visible in the training data; if a joint is not exercised, then the body parts it connects cannot be distinguished. Finally, our model requires that the observations arise from a scene containing roughly articulated figures; it would be a poor model of an octopus, for example.

An important direction for future study is the ability of learned skeletal structures

to generalize: applying them to new motions not seen during training, and to related sequences, such as using a model trained on one giraffe to parse the motion of another.

Chapter 5

Discussion and Future Directions

In this work we have described three projects that develop machine learning tools and apply them to the analysis of visual motion. In each, we took a challenging task which typically involved expert manual intervention, and presented a method by which the task could be solved automatically through the probabilistic modelling of data, resulting in systems that are more flexible and useful. First, we automated the collection of data required for training the appearance model of a visual tracker. This system leveraged a novel algorithm for learning PCA incrementally, incorporating new views of the target as they are acquired during tracking. Second, we demonstrated a technique for automating the selection of appearance and dynamics models best suited for a given tracking task. This relied on a new discriminative framework which, based on labelled sequences of the target, was able to robustly combine a number of weak models to produce a reliable tracker. Finally, we proposed a new model for automatically recovering the articulated skeletal structure—as well as its parameters and motion—from tracked feature points. The model can be fit in an entirely unsupervised fashion, automating the painstaking manual construction typically required in motion capture and tracking applications.

These projects have made a number of material contributions, but also suggest many directions where future research is required.

5.1 Remaining Challenges

5.1.1 Incremental Tracking

The proposed algorithm is able to track target objects through sequences of moderate duration. However, in practise it is limited by the fact that when it fails for more than a very brief interval, it is generally unable to recover the target. Part of this problem stems from the strategy used to incrementally update the appearance model. Currently, at each frame we simply select the maximum a posteriori subwindow and use it to re-fit the PCA model. An important remaining challenge is how to optimally make use of estimated uncertainty in the target's position during these updates. This could range from simply disregarding training subwindows when the positional uncertainty is too high, or the appearance likelihood is too low, to more-sophisticated schemes that weight the contribution of a new exemplar based on the degree of uncertainty. Furthermore, the tracker could be made more robust through the addition of a mechanism to detect when it has lost track of the target, and to broaden its field of search until the target is recovered.

5.1.2 Combining Discriminative Features

Although this project presents a promising probabilistic framework for modelling trajectories, the tracking experiments included currently constitute only a proof-of-concept. Further work, such as proper handling of scale and rotation, would be required to apply this tracker to more-interesting videos. Another limitation to the tracker's general applicability is the requirement of labelled training video. Obtaining the ground truth location of the target (4 parameters per frame) can be time-consuming, so it would be ideal if learning could be performed on partially labelled data. A final challenge is to modify the framework to allow a more-general set of features, such as higher-order or non-parametric dynamics, and incrementally adaptive appearance models.

5.1.3 Learning Skeletons

This project presents a model that is able to approximate skeletal structure, as well as parameters and motion, from a set of tracked feature point trajectories. Although it is robust to significant occlusion and missing data, it requires that the input trajectories be tracked somewhat more reliably than is possible with an off-the-shelf Kanade-Lucas-Tomasi feature tracker. One advantage of having an accurate articulated model of a non-rigid body is that it permits the locations of unobserved body features to be estimated. As such, if an articulated model of the target were available a priori, it could be leveraged to significantly improve feature tracking performance. This suggests an important direction for future research: perhaps performance of both tracking and articulated structure learning could be improved by integrating the two tasks.

Another disadvantage of the model is its determination to recover a single best structure. Since an articulated stick figure is at best an approximation—at a finer level of detail the human body, for example, exhibits signifiantly more non-rigidity—capturing the uncertainty in the stick figure's structure might produce a more-accurate model.

Some preliminary work along these lines, applying nonparametric Bayesian methods for learning tree structures, appears in (Meeds et al., 2008). This approach proposes to represent uncertainty in the structure and model parameters via samples from the posterior distribution given the observation sequence. Despite the theoretical advantages the approach presents a number of implementation challenges, including the difficulty of developing a Markov chain Monte Carlo method which can reliably simulate samples from the posterior, as well as the need to manually tune a large number of unintuitive hyper-parameters. In practice, it has been difficult to use this approach to recover even 2D articulated structure from 2-dimensional planar motion, and applying it to the challenging sequences presented in Chapter 4 does not seem feasible at this point.

5.1.4 Learning to Analyze Visual Motion

Finally, when considering the general problem of building computer models to analyze and understand visual motion, the work here represents only a small step. Rather than simply automating individual parts of hand-built vision systems, the ultimate goal is to develop end-to-end learning systems that map directly from raw inputs to semantic representations. Such approaches, with limited opportunity to benefit from expert domain knowledge, might seem to be at a disadvantage. However the current ubiquity of digital cameras provides a limitless source of data, and the growth of computer power continues unabated. Thus developing scalable automated algorithms is the key to building new computer systems with increased flexibility and coverage.

Appendix A

The Multivariate Gaussian Probability Distribution

A.1 Parametrization and Multiplication

Throughout this work we model the probability of real vector-valued random variables, $\mathbf{x} \in \mathbb{R}^D$, using the multivariate Gaussian (or Normal) distribution. The probability density of such a variable is written as a function of two parameters, most commonly the *mean vector* $\boldsymbol{\mu} \in \mathbb{R}^D$ and the positive-definite *covariance matrix* $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$. Since these two parameters are the first and second moments of the distribution, this is known as the *moment parametrization*. The resulting density function is

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right).$$

A well-known fact about Gaussian distributions is that the product of a number of Gaussians, each with its own mean and covariance, is proportional to another Gaussian.

Many of facts presented in this appendix are described in much greater detail by Jordan (200X). Regrettably, at this time Jordan's excellent book is available only as an unpublished manuscript. Other sources for this information include (Roweis, 1999a,b; Petersen and Pedersen, 2008).

Algebraically:

$$\prod_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \propto \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\text{where } \boldsymbol{\Sigma} = \left(\sum_i \boldsymbol{\Sigma}_i^{-1}\right)^{-1} \quad \text{and} \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\sum_i \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_k\right)$$

Deriving this result can be cumbersome, since it involves expanding the quadratic inside the exponential of each Gaussian, summing the quadratics, and then completing the square of the result (see *Completing the Square*, Section A.5). Furthermore, implementing this product operation can be computationally expensive, due to the number of matrix inverses required.

Consider however the following alternative. Suppose the quadratic term appearing in the exponential were expanded, and this parametrization of the density function was used instead. Then multiplying Gaussians would be a simple matter of adding the quadratics, no expanding and completing the square required. The resulting density function would look as follows:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\nu}, \boldsymbol{\tau}) = \exp\left(-\frac{1}{2}\mathbf{x}^\top\boldsymbol{\tau}\mathbf{x} + \mathbf{x}^\top\boldsymbol{\nu} + c\right).$$

where $c = -\frac{1}{2}(D\log(2\pi) - |\boldsymbol{\tau}| + \boldsymbol{\nu}^\top\boldsymbol{\tau}\boldsymbol{\nu})$ is a normalization constant. This parametrization of the Gaussian density is known as the *canonical parametrization*, and is characterized by a *precision matrix* $\boldsymbol{\tau}$, and a *scaled-mean vector* $\boldsymbol{\nu}$. As expected, in this form the product operation is trivial:

$$\prod_i \mathcal{N}(\mathbf{x}|\boldsymbol{\nu}_i, \boldsymbol{\tau}_i) \propto \mathcal{N}(\mathbf{x}|\boldsymbol{\nu}, \boldsymbol{\tau}) \quad \text{where} \quad \boldsymbol{\tau} = \sum_i \boldsymbol{\tau}_i \quad \text{and} \quad \boldsymbol{\nu} = \sum_i \boldsymbol{\nu}_i$$

The relationship between the canonical parameters and the moments is quite simple; conversion between the two forms can be achieved using the following identities:

$$\boldsymbol{\Sigma} = \boldsymbol{\tau}^{-1} \quad \text{and} \quad \boldsymbol{\mu} = \boldsymbol{\tau}^{-1}\boldsymbol{\nu}.$$

A.2 Marginalization

Another important operation performed on multivariate random variables is marginalization. Suppose we have a multivariate Gaussian random variable $\mathbf{x}$, whose elements can be partitioned into to sub-vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Using block matrix notation, in the moment parametrization this can be expressed as

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \equiv \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right). \tag{A.1}$$

The goal of marginalization is, given the probability density of $\mathbf{x}$, to compute the density of $\mathbf{x}_2$ irrespective of $\mathbf{x}_1$. This "marginal" density is obtained by integrating $\mathbf{x}_1$ out of the "joint" density: $\mathrm{P}(\mathbf{x}_2) = \int P(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1$. (The marginal probability of $\mathbf{x}_1$ can be obtained analogously.)

In the moment parametrization, marginalization is trivial. The marginal density of $\mathbf{x}_2$ is simply a multivariate Gaussian with mean $\boldsymbol{\mu}_2$ and covariance $\boldsymbol{\Sigma}_{22}$ (Petersen and Pedersen, 2008):

$$\int \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right) dx_1 = \mathcal{N}(\mathbf{x}_2|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}).$$

In the canonical parametrization, however, this operation is significantly more challenging. In fact, the simplest way to obtain the marginal precision and scaled-mean is to convert from the canonical to moment parametrization, take the marginal mean and covariance, and covert them back to canonical form. Suppose our Gaussian density can be partitioned as follows:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\nu}, \boldsymbol{\tau}) \equiv \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\nu_1 \\ \nu_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22}\end{bmatrix}\right) \tag{A.2}$$

To obtain the marginal precision we must first invert the (joint) precision matrix, using

identities for inverting block matrices:

$$\begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\tau}_{11}^{-1} + \boldsymbol{\tau}_{11}^{-1}\boldsymbol{\tau}_{12}\mathbf{P}^{-1}\boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1} & -\boldsymbol{\tau}_{11}^{-1}\boldsymbol{\tau}_{21}\mathbf{P}^{-1} \\ -\mathbf{P}^{-1}\boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1} & \mathbf{P}^{-1} \end{bmatrix}, \tag{A.3}$$

where $\mathbf{P} = \boldsymbol{\tau}_{22} - \boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1}\boldsymbol{\tau}_{12}$ is the Schur complement (Petersen and Pedersen, 2008). Thus the marginal precision for $\mathbf{x}_2$ is $\mathbf{P}$. The marginal scaled-mean is obtained by multiplying the scaled-mean $(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2)$ by the inverse precision (A.3), selecting the corresponding rows, then multiplying by the precision $\mathbf{P}$:

$$\mathbf{P}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}^{-1}\begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \mathbf{P}(-\mathbf{P}^{-1}\boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1}\boldsymbol{\nu}_1 + \mathbf{P}^{-1}\boldsymbol{\nu}_2) = \boldsymbol{\nu}_2 - \boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1}\boldsymbol{\nu}_1$$

Putting this all together, the marginal density of $\mathbf{x}_2$ is

$$\int \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \mid \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}\right) d\mathbf{x}_1 = \mathcal{N}(\mathbf{x}_2 |\, \boldsymbol{\nu}_2 - \boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1}\boldsymbol{\nu}_1,\ \boldsymbol{\tau}_{22} - \boldsymbol{\tau}_{21}\boldsymbol{\tau}_{11}^{-1}\boldsymbol{\tau}_{12}).$$

A.3 Conditioning

The third operation we will consider is conditioning. Given the aforementioned partition of Gaussian random variable $\mathbf{x}$ into $\mathbf{x}_1$ and $\mathbf{x}_2$, the goal of conditioning is to compute the conditional probability $\mathrm{P}(\mathbf{x}_1|\mathbf{x}_2)$ of $\mathbf{x}_1$ given a particular value of $\mathbf{x}_2$. (The conditional probability of $\mathbf{x}_2$ given $\mathbf{x}_1$ can be obtained analogously.)

Unlike marginalization, conditioning is much simpler in the canonical parametrization. Beginning with the partitioned form of the canonical density, equation (A.2), the conditional density of $\mathbf{x}_1$ is also a Gaussian $\mathrm{P}(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\boldsymbol{\nu}_1 - \boldsymbol{\tau}_{12}\mathbf{x}_2, \boldsymbol{\tau}_{11})$. This result can be obtained by simply rearranging the terms inside the exponential

$$\begin{aligned} &-\frac{1}{2}\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} \end{bmatrix}\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix} \\ &\quad = -\frac{1}{2}(\mathbf{x}_1{}^\top\boldsymbol{\tau}_{11}\mathbf{x}_1 + 2\mathbf{x}_1{}^\top\boldsymbol{\tau}_{12}\mathbf{x}_2 + \mathbf{x}_2{}^\top\boldsymbol{\tau}_{22}\mathbf{x}_2) + \mathbf{x}_1{}^\top\boldsymbol{\nu}_1 + \mathbf{x}_2{}^\top\boldsymbol{\nu}_2 \\ &\quad = -\frac{1}{2}\mathbf{x}_1{}^\top\boldsymbol{\tau}_{11}\mathbf{x}_1 + \mathbf{x}_1{}^\top(\boldsymbol{\nu}_1 - \boldsymbol{\tau}_{12}\mathbf{x}_2) + \text{terms constant in } \mathbf{x}_1. \end{aligned}$$

The identity for conditioning in the moment parametrization can be obtained by converting the canonical result. Assume we begin with a moment-parametrization density written in block form, as in equation (A.1). The joint covariance can be inverted using another identity for block matrices

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{C}^{-1} & -\mathbf{C}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1} \\ -\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\mathbf{C}^{-1} & \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12} \end{bmatrix}$$

where $\mathbf{C} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}$ (Petersen and Pedersen, 2008). Since the conditional precision is equal to the $\boldsymbol{\tau}_{11}$ block of the joint precision, we can see that the conditional covariance is $\mathbf{C}$.

To convert the conditional scaled-mean $\boldsymbol{\nu}_1 - \boldsymbol{\tau}_{12}\mathbf{x}_2$ to the conditional mean we must solve for $\boldsymbol{\nu}_1$ and $\boldsymbol{\tau}_{12}$. From the block-inversion of the covariance, we note that $\boldsymbol{\tau}_{12} = -\mathbf{C}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}$ and

$$\boldsymbol{\nu}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} = \mathbf{C}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\mu}_2).$$

Putting these together, we note the conditional mean is

$$\begin{aligned} \mathbf{C}(\boldsymbol{\nu}_1 - \boldsymbol{\tau}_{12}\mathbf{x}_2) &= \mathbf{C}\mathbf{C}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\mu}_2) + \mathbf{C}\mathbf{C}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\mathbf{x}_2 \\ &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2). \end{aligned}$$

Therefore, in the moment parametrization, the conditional density of $\mathbf{x}_1$ given $\mathbf{x}_2$ is $\mathrm{P}(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\, \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2),\, \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21})$.

A.4 Expectation of a Quadratic

The final operation we will consider is computing the expected value of a quadratic form, $(\mathbf{x} - \mathbf{y})^\top(\mathbf{x} - \mathbf{y})$, when variable $\mathbf{x}$ is distributed according to a multivariate Gaussian

$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The result of this operation is (Roweis, 1999b; Petersen and Pedersen, 2008):

$$\begin{aligned}\mathrm{E}[(\mathbf{x}-\mathbf{y})^\top(\mathbf{x}-\mathbf{y})] &= \int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})(\mathbf{x}-\mathbf{y})^\top(\mathbf{x}-\mathbf{y})d\mathbf{x} \\ &= (\boldsymbol{\mu}-\mathbf{y})^\top(\boldsymbol{\mu}-\mathbf{y}) + \mathrm{tr}(\boldsymbol{\Sigma}).\end{aligned}$$

A.5 Completing the Square

When discussing the product of Gaussian densities, we noted that the derivation relies on *Completing the Square*. Although well known in scalar terms, this identity can also be applied to quadratics that contain vector-valued variables:

$$\begin{aligned}&\sum_i (\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) \\ &\quad = (\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \; - \boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \; + \sum_i \boldsymbol{\mu}_i{}^\top\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\end{aligned}$$

where $\boldsymbol{\Sigma} \equiv (\sum_i \boldsymbol{\Sigma}_i^{-1})^{-1}$ and $\boldsymbol{\mu} \equiv \boldsymbol{\Sigma}(\sum_i \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i)$. The proof of this result is as follows:

$$\begin{aligned}&\sum_i (\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) \\ &\quad = \mathbf{x}^\top\left(\sum_i \boldsymbol{\Sigma}_i^{-1}\right)\mathbf{x} - 2\mathbf{x}^\top\left(\sum_i \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\right) + \sum_i \boldsymbol{\mu}_i{}^\top\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i \\ &\quad = \mathbf{x}^\top\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\mathbf{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \sum_i \boldsymbol{\mu}_i{}^\top\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i \\ &\quad = (\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) - \boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \sum_i \boldsymbol{\mu}_i{}^\top\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i.\end{aligned}$$

Appendix B

Combining Discriminative Features Derivation: Inference and Learning

In this appendix we derive the inference and learning algorithms for the *Combining Discriminative Features* model described in Chapter 3. We begin by deriving exact inference and learning for the simpler linear-Gaussian case, in which all features are active. Then we extend this solution to the more-interesting non-linear case by introducing switching variables, and derive approximate inference and learning algorithms.

The training data is a set of N sequences pairs, $\{(\mathbf{X}^n, \mathbf{Y}^n)\}_{n=1}^N$, where $\mathbf{X}^n$ and $\mathbf{Y}^n$ are the state sequence and observation sequence respectively, and from this we construct a model of the conditional distribution of $\mathbf{X}$ given $\mathbf{Y}$. This differs from generative approaches, which model $\mathbf{X}|\mathbf{Y}$ indirectly by first specifying a prior over $\mathbf{X}$, fitting a likelihood distribution $\mathbf{Y}|\mathbf{X}$, and applying Bayes' rule. The conditional distribution is formed by taking a log-linear combination of features. The features are f_j and g_k, which are linearly related to weights $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_k$ respectively. The observation features g_k relate the state at a single time-step to the observations, while the dynamics features f_j involve pairs of states at adjacent time-steps, capturing the evolution of the sequence. Finally, the distribution is normalized by the *partition function* $Z(\mathbf{Y})$:

$$\mathrm{P}(\mathbf{X}^n|\mathbf{Y}^n) = \frac{1}{Z(\mathbf{Y}^n)} \exp\left(\sum_{t=2}^{\top}\sum_{j=1}^{J} f_j(\mathbf{x}_{t-1}^n, \mathbf{x}_t^n) + \sum_{t=1}^{\top}\sum_{k=1}^{K} g_k(\mathbf{x}_t^n, \mathbf{Y}^n)\right).$$

B.1 Linear Dynamics with Gaussian Noise

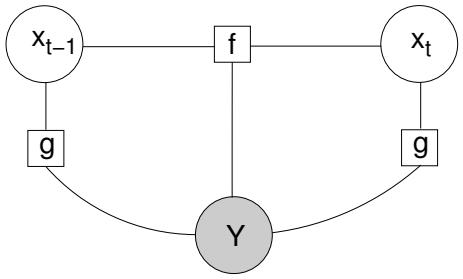

Figure B.1: The factor graph of the linear Gaussian model.

In the non-switching case exact inference, and therefore exact learning, is possible. All dynamics features propose that the state at any time is a linear function of the previous state, plus Gaussian noise. That is $\mathbf{x}_t = \mathbf{T}_j\mathbf{x}_{t-1}+\mathbf{d}_j+\mathcal{N}(0, \boldsymbol{\alpha}_j)$ for some transition matrix $\mathbf{T}_j$, translation $\mathbf{d}_j$, and precision[1]. As presented in Chapter 3, the feature functions used are

$$f_j(\mathbf{x}_{t-1}, \mathbf{x}_t) = -\frac{1}{2}\left(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1})\right)^{\top}\boldsymbol{\alpha}_j\left(\mathbf{x}_t - \phi_j(\mathbf{x}_{t-1})\right)$$
$$g_k(\mathbf{x}_t, \mathbf{Y}) = -\frac{1}{2}\left(\mathbf{x}_t - \gamma_k(\mathbf{Y}, t)\right)^{\top}\boldsymbol{\beta}_k\left(\mathbf{x}_t - \gamma_k(\mathbf{Y}, t)\right),$$

where $\phi_j(\mathbf{x}_{t-1})$ predicts the state at time t using a linear dynamics $\phi_j(\mathbf{x}_{t-1}) = \mathbf{T}_j\mathbf{x}_{t-1}+\mathbf{d}_j$, and $\gamma_k(\mathbf{Y}, t)$ is a prediction of the state at time t obtained by considering any or all of the observations.

[1]Throughout this appendix, except where noted, we will parametrize the Gaussian distribution using the *canonical parametrization*, as described in Appendix A. Thus $\mathcal{N}(\mathbf{x}|\boldsymbol{\nu}, \boldsymbol{\tau})$ has a precision matrix $\boldsymbol{\tau}$, which is simply the inverse of the covariance matrix $\boldsymbol{\tau} = \boldsymbol{\Sigma}^{-1}$, and a mean given by $\boldsymbol{\tau}^{-1}\boldsymbol{\nu}$.

B.1.1 Inference

Inference in this model is the process of computing the probability distribution over the state sequence $\mathbf{X}$ given an observation sequence $\mathbf{Y}$, $\mathrm{P}(\mathbf{X}|\mathbf{Y})$. Since all features are quadratic functions in $\mathbf{X}$, the resulting distribution is Gaussian, and exact inference can be performed efficiently using the belief propagation algorithm (Jordan, 200X). Inference in this model is closely linked, and in some ways equivalent, to the Kalman smoother (Rauch et al., 1965). Inference via BP allows us to: (1) compute the marginal $\mathrm{P}(\mathbf{x}_t|\mathbf{Y})$ and pairwise marginal $\mathrm{P}(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{Y})$ distributions, required for learning, (2) draw samples from the distribution, and (3) compute the mode, or most-probably state sequence, $\mathrm{argmax}_{\mathbf{X}} \mathrm{P}(\mathbf{X}|\mathbf{Y})$.

Implementing belief propagation requires recursively computing three types of message: $m_{t-1,t}(\mathbf{x}_t)$, which passes information from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$; $m_{t+1,t}(\mathbf{x}_t)$ which sends information in the reverse direction; and $m_t(\mathbf{x}_t)$ which sends information to $\mathbf{x}_t$ from the observations. In addition to these messages we compute two intermediate results: $fwd(\mathbf{x}_t)$ and $bwd(\mathbf{x}_t)$. These correspond respectively to the forward-filtered estimate of $\mathbf{x}_t$, relying only on evidence provided from observation features at time t and earlier, and the backward-filtered estimate of $\mathbf{x}_t$, relying on observation features from time t and later.

Before deriving the messages, we make note of a useful fact. Consider the contribution to the conditional probability made by the pairwise dynamics features,

$$\Psi(\mathbf{x}_{t-1}, \mathbf{x}_t) = \exp\left(-\frac{1}{2}\sum_{j=1}^{J}\left(\mathbf{x}_t - \mathbf{T}_j\mathbf{x}_{t-1} - \mathbf{d}_j\right)^{\top}\boldsymbol{\alpha}_j\left(\mathbf{x}_t - \mathbf{T}_j\mathbf{x}_{t-1} - \mathbf{d}_j\right)\right).$$

By expanding the quadratic and collecting terms, we see that this factor is a product of Gaussian distributions, and thus is proportional to a single Gaussian, expressed in canonical parametrization:

$$\Psi(\mathbf{x}_{t-1}, \mathbf{x}_t) \propto \exp\left(-\frac{1}{2}\begin{bmatrix}\mathbf{x}_{t-1}\\ \mathbf{x}_t\end{bmatrix}^{\top}\begin{bmatrix}T\alpha T & -\alpha T^{\top}\\ -\alpha T & \mathbf{A}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{t-1}\\ \mathbf{x}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_{t-1}\\ \mathbf{x}_t\end{bmatrix}^{\top}\begin{bmatrix}-T\alpha d\\ \alpha d\end{bmatrix}\right).$$

(Details on products of Gaussians and the canonical parametrization are included in Appendix A.) The above expression makes use of the following shorthand symbols which, for the linear-Gaussian case, need not be recomputed at each time-step:

$$\mathbf{A} = \sum_j \boldsymbol{\alpha}_j \qquad \alpha T = \sum_j \alpha_j \mathbf{T}_j$$

$$T\alpha T = \sum_j \mathbf{T}_j{}^\top \boldsymbol{\alpha}_j \mathbf{T}_j \qquad \alpha d = \sum_j \boldsymbol{\alpha}_j \mathbf{d}_j$$

$$T\alpha d = \sum_j \mathbf{T}_j \boldsymbol{\alpha}_j \mathbf{d}_j.$$

Forward Message Passing (Filtering)

The forward propagation of the state estimates, useful for online tracking, is related to Kalman filtering. To predict $\mathbf{x}_t$ given the observation features $\gamma_k(\mathbf{Y}, 1:t-1)$, the prior (corrected) state estimate of $\mathbf{x}_{t-1}|\gamma_k(\mathbf{Y}, 1:t-1)$, $fwd(\mathbf{x}_{t-1})$, must be combined with the dynamics factor $\Psi(\mathbf{x}_{t-1}, \mathbf{x}_t)$. Since both factors are Gaussian in $(\mathbf{x}_{t-1}, \mathbf{x}_t)$, computing the message can be performed by multiplying the two Gaussians and marginalizing out $\mathbf{x}_{t-1}$.

$$\begin{aligned} m_{t-1,t}(\mathbf{x}_t) &\propto \int_{\mathbf{x}_{t-1}} \Psi(\mathbf{x}_{t-1}, \mathbf{x}_t) fwd(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \\ &\propto \int_{\mathbf{x}_{t-1}} \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_t \end{bmatrix} \mid \begin{bmatrix} -T\alpha d \\ \alpha d \end{bmatrix}, \begin{bmatrix} T\alpha T & -\alpha T^\top \\ -\alpha T & \mathbf{A} \end{bmatrix} \right) \times \\ &\qquad \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_t \end{bmatrix} \mid \begin{bmatrix} \boldsymbol{\mu}^f_{t-1} \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\tau}^f_{t-1} & 0 \\ 0 & 0 \end{bmatrix} \right) d\mathbf{x}_{t-1}. \end{aligned}$$

Upon performing the multiplication and marginalization (details of these operations are found in Appendix A), the resulting message, the prediction of $\mathbf{x}_t|\gamma_k(\mathbf{Y}, 1:t-1)$, is also a Gaussian $m_{t-1,t}(\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_t|\boldsymbol{\nu}_{t|t-1}, \boldsymbol{\tau}_{t|t-1}\right)$, where:

$$\boldsymbol{\tau}_{t|t-1} = \mathbf{A} - \alpha T (T\alpha T + \boldsymbol{\tau}^f_{t-1})^{-1} \alpha T^\top$$

$$\boldsymbol{\nu}_{t|t-1} = \alpha T (T\alpha T + \boldsymbol{\tau}^f_{t-1})^{-1} (\boldsymbol{\tau}^f_{t-1} \boldsymbol{\mu}^f_{t-1} - T\alpha d) + \alpha d.$$

This expression for $\boldsymbol{\tau}_{t|t-1}$ has the potential to exhibit numerical instabilities, since it is the difference between two positive semi-definite matrices, which is not guaranteed to

produce a positive semi-definite result (Welling, "The Kalman Filter"). An equivalent alternative expression for $\boldsymbol{\tau}_{t|t-1}$, related by the Sherman-Morrison-Woodbury formula (Petersen and Pedersen, 2008) is

$$\boldsymbol{\tau}_{t|t-1} = (\mathbf{A}^{-1} + \mathbf{A}^{-1}\alpha T(T\alpha T + \boldsymbol{\tau}_{t-1}^{f} - \alpha T^{\top}\mathbf{A}^{-1}\alpha T)^{-1}\alpha T^{\top}\mathbf{A}^{-1})^{-1}.$$

Despite incurring additional computational cost, in practice this expression produces a more numerically reliable result.

Incorporating Evidence

Updating the state estimate to include information from the observation features is a simple matter of combining $m_{t-1,t}(\mathbf{x}_t)$ with the message sent from the observations, which will be denoted as $m_t(\mathbf{x}_t)$. The message from the observation features to the hidden state is a product of K Gaussian distributions, each of the form $\mathcal{N}(\mathbf{x}_t|\boldsymbol{\beta}_k\gamma_k(\mathbf{Y},t),\boldsymbol{\beta}_k)$. Thus the resulting message is also Gaussian, of the form $m_t(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t|\mathbf{y}'_t,\mathbf{B})$, with $\mathbf{B} = \sum_{k=1}^{K}\boldsymbol{\beta}_k$ and $\mathbf{y}'_t = (\sum_{k=1}^{K}\boldsymbol{\beta}_k\gamma_k(\mathbf{Y},t))$.

Combining the forward message with the message from the observations is again a simple product of Gaussians: $fwd(\mathbf{x}_t) \propto m_{t-1,t}(\mathbf{x}_t)\, m_t(\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_t|\boldsymbol{\mu}_t^f, \boldsymbol{\tau}_t^f\right)$, where:

$$\boldsymbol{\tau}_t^f = \boldsymbol{\tau}_{t|t-1} + \mathbf{B}$$
$$\boldsymbol{\nu}_t^f = \boldsymbol{\nu}_{t|t-1} + \mathbf{y}'_t.$$

Backward message passing (Smoothing)

The backward propagation of the state estimate can be performed analogously. The backward prediction of $\mathbf{x}_t$ given $\gamma_k(\mathbf{Y}, t+1:T)$, $m_{t+1,t}(\mathbf{x}_t)$, can be expressed as

$$\begin{aligned} m_{t+1,t}(\mathbf{x}_t) &\propto \int_{\mathbf{x}_{t+1}} \Psi(\mathbf{x}_t, \mathbf{x}_{t+1}) bwd(\mathbf{x}_{t+1}) d\mathbf{x}_{t+1} \\ &\propto \int_{\mathbf{x}_{t+1}} \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t+1} \end{bmatrix} \mid \begin{bmatrix} -T\alpha d \\ \alpha d \end{bmatrix}, \begin{bmatrix} T\alpha T & -\alpha T^\top \\ -\alpha T & \mathbf{A} \end{bmatrix} \right) \times \\ &\qquad \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t+1} \end{bmatrix} \mid \begin{bmatrix} 0 \\ \boldsymbol{\mu}^b_{t+1} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{\tau}^b_{t+1} \end{bmatrix} \right) d\mathbf{x}_{t+1}. \end{aligned}$$

The resulting message is $m_{t+1,t}(\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_t | \boldsymbol{\nu}_{t|t+1}, \boldsymbol{\tau}_{t|t+1}\right)$, where

$$\begin{aligned} \boldsymbol{\tau}_{t|t+1} &= T\alpha T - \alpha T^\top (\mathbf{A} + \boldsymbol{\tau}^b_{t+1})^{-1} \alpha T \\ \boldsymbol{\nu}_{t|t+1} &= \alpha T^\top (\mathbf{A} + \boldsymbol{\tau}^b_{t+1})^{-1} (\boldsymbol{\tau}^b_{t+1} \boldsymbol{\mu}^b_{t+1} + \alpha d) - T\alpha d. \end{aligned}$$

As with forward prediction, an equivalent alternative expression for the prediction that avoids taking the difference of positive definite matrices is

$$\boldsymbol{\tau}_{t|t+1} = ((T\alpha T)^{-1} + (T\alpha T)^{-1} \alpha T^\top (\mathbf{A} + \boldsymbol{\tau}^b_{t+1} - \alpha T (T\alpha T)^{-1} \alpha T^\top)^{-1} \alpha T (T\alpha T)^{-1})^{-1}.$$

Incorporating evidence from the observations uses the same message computed for the forward pass. This gives, as a corrected backward prediction: $bwd(\mathbf{x}_t) \propto m_{t,t+1}(\mathbf{x}_t)\, m_t(\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_t | \boldsymbol{\nu}^b_t, \boldsymbol{\tau}^b_t\right)$ where

$$\begin{aligned} \boldsymbol{\tau}^b_t &= \boldsymbol{\tau}_{t|t+1} + \mathbf{B} \\ \boldsymbol{\nu}^b_t &= \boldsymbol{\nu}_{t|t+1} + \mathbf{y}'_t. \end{aligned}$$

Marginals, Pairwise Marginals, and Mode

The marginal distributions, $\mathrm{P}(\mathbf{x}_t|\mathbf{Y})$, required for learning, can be computed by multiplying together all messages coming into $\mathbf{x}_t$. This is a Gaussian product $\mathcal{N}(\boldsymbol{\nu}_t, \boldsymbol{\tau}_t) \propto$

$m_{t-1,t}(\mathbf{x}_t)\, m_t(\mathbf{x}_t)\, m_{t+1,t}(\mathbf{x}_t)$, with

$$\boldsymbol{\tau}_t = \boldsymbol{\tau}_{t|t-1} + \mathbf{B} + \boldsymbol{\tau}_{t|t+1}$$

$$\boldsymbol{\nu}_t = \boldsymbol{\nu}_{t|t-1} + \mathbf{y}'_t + \boldsymbol{\nu}_{t|t+1},$$

or in moment parametrization,

$$\boldsymbol{\Sigma}_t = (\boldsymbol{\tau}_{t|t-1} + \mathbf{B} + \boldsymbol{\tau}_{t|t+1})^{-1}$$

$$\boldsymbol{\mu}_t = \boldsymbol{\Sigma}_t(\boldsymbol{\nu}_{t|t-1} + \mathbf{y}'_t + \boldsymbol{\nu}_{t|t+1}).$$

Similarly, the pairwise marginal for $(\mathbf{x}_t, \mathbf{x}_{t+1})$ can be computed by combining all messages coming into the pair (forward, backward, and observation), along with the dynamics factor:

$$\begin{aligned} &m_{t-2,t-1}(\mathbf{x}_{t-1})\, m_{t-1}(\mathbf{x}_{t-1})\, \Psi(\mathbf{x}_{t-1}, \mathbf{x}_t)\, m_{t,t+1}(\mathbf{x}_t)\, m_t(\mathbf{x}_t) \\ &\qquad = fwd(\mathbf{x}_{t-1})\, \Psi(\mathbf{x}_{t-1}, \mathbf{x}_t)\, bwd(\mathbf{x}_{t+1}). \end{aligned}$$

The mean of the pairwise marginal is simply the concatenation of the marginal means, $(\boldsymbol{\mu}_{t-1}, \boldsymbol{\mu}_t)$. The pairwise marginal precision is sum of the precisions from the messages and dynamics factor

$$\boldsymbol{\tau}_{pair(t-1,t)} = \begin{bmatrix} \mathbf{B} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\tau}_{t-1|t-2} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} T\alpha T & -\alpha T^\top \\ -\alpha T & \mathbf{A} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{\tau}_{t|t+1} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{B} \end{bmatrix}.$$

The marginal cross covariance, $\boldsymbol{\Sigma}_{t-1,t} = \mathrm{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_t - \boldsymbol{\mu}_t)^\top]$ can be obtained by inverting the pairwise marginal precision and selecting the upper-right block of the result.

Since the distribution of $\mathbf{X}$ given $\mathbf{Y}$ is Gaussian, the mode, or most likely sequence $\boldsymbol{\mu} \equiv \mathrm{argmax}_{\mathbf{X}}\, \mathrm{P}(\mathbf{X}|\mathbf{Y})$, is simply the concatenation of marginal means: $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_T)$.

Sampling

After performing inference to obtain the marginals and pairwise marginals, drawing a sample from $\mathrm{P}(\mathbf{X}|\mathbf{Y})$ is a simple iterative procedure. First, begin by drawing a sample of the state at time T from its marginal:

$$\hat{\mathbf{x}}_t \sim \mathcal{N}(\boldsymbol{\nu}_t, \boldsymbol{\tau}_T).$$

Then for each time t, starting at $T-1$ and working backwards, draw a sample $\hat{\mathbf{x}}_t$ from the conditional distribution $\mathrm{P}(\mathbf{x}_t|\hat{\mathbf{x}}_{t+1})$, given the sample that was just drawn for time $t+1$. The conditional can easily be obtained from the pairwise marginal distribution $\mathrm{P}(\mathbf{x}_t, \mathbf{x}_{t+1}|\mathbf{Y})$ using the conditioning formula presented in Appendix A.3:

$$\hat{\mathbf{x}}_t|\hat{\mathbf{x}}_{t+1} \sim \mathcal{N}(\boldsymbol{\nu}_t - \boldsymbol{\tau}_{t,t+1}\hat{\mathbf{x}}_{t+1},\ \boldsymbol{\tau}_{t,t}).$$

A similar forward sampling procedure can also be derived, beginning with a sample from $\mathrm{P}(\mathbf{x}_1)$ and employing $\mathrm{P}(\mathbf{x}_t|\hat{\mathbf{x}}_{t-1})$.

B.1.2 Learning

Given a set of N labelled sequences, learning the $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}_k$ parameters in this model is accomplished via gradient ascent on the log-likelihood:

$$\begin{aligned}
\mathcal{L} &= \sum_{n=1}^{N}\Bigg(\sum_{t=2}^{\top}\sum_{j=1}^{J} -\frac{1}{2}\left(\mathbf{x}_t^n - \mathbf{T}_j\mathbf{x}_{t-1}^n\right)^{\top}\boldsymbol{\alpha}_j\left(\mathbf{x}_t^n - \mathbf{T}_j\mathbf{x}_{t-1}^n\right) \\
&\quad + \sum_{t=1}^{\top}\sum_{k=1}^{K} -\frac{1}{2}\left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right)^{\top}\boldsymbol{\beta}_k\left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right) - \log Z(\mathbf{Y}^n)\Bigg) \\
&= \sum_{n=1}^{N}\Bigg(\sum_{j=1}^{J}\mathrm{tr}\left(\mathbf{F}_j^n\boldsymbol{\alpha}_j\right) + \sum_{k=1}^{K}\mathrm{tr}\left(\mathbf{G}_k^n\boldsymbol{\beta}_k\right) - \log Z(\mathbf{Y}^n)\Bigg),
\end{aligned}$$

where we define shorthands $\mathbf{F}_j^n$ and $\mathbf{G}_k^n$ as

$$\begin{aligned}
\mathbf{F}_j^n &= -\frac{1}{2}\sum_{t=2}^{\top}\left(\mathbf{x}_t^n - \mathbf{T}_j\mathbf{x}_{t-1}^n\right)\left(\mathbf{x}_t^n - \mathbf{T}_j\mathbf{x}_{t-1}^n\right)^{\top}, \\
\mathbf{G}_k^n &= -\frac{1}{2}\sum_{t=1}^{\top}\left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right)\left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right)^{\top}.
\end{aligned}$$

Taking derivatives gives us:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}_j} &= \sum_{n=1}^{N}\left(\mathbf{F}_j^n - \mathrm{E}_{\mathrm{P}(\mathbf{X}|\mathbf{Y}^n)}\left[\mathbf{F}_j^n\right]\right) \\
&= \sum_{n=1}^{N}\left(\mathbf{F}_j^n + \frac{1}{2}\sum_{t=2}^{\top}\mathrm{E}_{\mathrm{P}(\mathbf{x}_{t-1}^n, \mathbf{x}_t^n|\mathbf{Y}^n)}\left[\left(\mathbf{x}_t^n - \mathbf{T}_j\mathbf{x}_{t-1}^n\right)\left(\mathbf{x}_t^n - \mathbf{T}_j\mathbf{x}_{t-1}^n\right)^{\top}\right]\right),
\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}_k} &= \sum_{n=1}^{N} \left(\mathbf{G}_k^n - \mathrm{E}_{\mathrm{P}(\mathbf{X}|\mathbf{Y}^n)}\left[\mathbf{G}_k^n\right]\right) \\ &= \sum_{n=1}^{N} \left(\mathbf{G}_k^n + \frac{1}{2} \sum_{t=1}^{\top} \mathrm{E}_{\mathrm{P}(\mathbf{x}_t^n|\mathbf{Y}^n)} \left[\left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right) \left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right)^{\top} \right] \right).\end{aligned}$$

The expectation term in $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_j}$ expands to

$$\begin{aligned}&\sum_{t=2}^{\top} \left(\mathrm{E}[\mathbf{x}_t^n \mathbf{x}_t^{n\top}] - \mathbf{T}_j \mathrm{E}[\mathbf{x}_{t-1}^n \mathbf{x}_t^{n\top}] - \mathrm{E}[\mathbf{x}_t^n \mathbf{x}_{t-1}^{n\top}] \mathbf{T}_j{}^{\top} + \mathbf{T}_j \mathrm{E}[\mathbf{x}_{t-1}^n \mathbf{x}_{t-1}^{n\top}] \mathbf{T}_j{}^{\top} \right) \\ &= \sum_{t=2}^{\top} \Bigg(\left(\boldsymbol{\Sigma}_t^n + \boldsymbol{\mu}_t^n \boldsymbol{\mu}_t^{n\top} \right) - \mathbf{T}_j \left(\boldsymbol{\Sigma}_{t-1,t}^n + \boldsymbol{\mu}_{t-1}^n \boldsymbol{\mu}_t^{n\top} \right) \\ &\quad - \left(\boldsymbol{\Sigma}_{t-1,t}^n + \boldsymbol{\mu}_{t-1}^n \boldsymbol{\mu}_t^{n\top} \right)^{\top} \mathbf{T}_j{}^{\top} + \mathbf{T}_j \left(\boldsymbol{\Sigma}_{t-1}^n + \boldsymbol{\mu}_{t-1}^n \boldsymbol{\mu}_{t-1}^{n\top} \right) \mathbf{T}_j{}^{\top} \Bigg).\end{aligned}$$

Similarly, the expectation term in $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}_k}$ expands to

$$\begin{aligned}&\sum_{t=1}^{\top} \left(\mathrm{E}[\mathbf{x}_t^n \mathbf{x}_t^{n\top}] - \gamma_k(\mathbf{Y}^n, t) \mathrm{E}[\mathbf{x}_t^n]^{\top} - \mathrm{E}[\mathbf{x}_t^n] \gamma_k(\mathbf{Y}^n, t)^{\top} + \gamma_k(\mathbf{Y}^n, t) \gamma_k(\mathbf{Y}^n, t)^{\top} \right) \\ &= \sum_{t=1}^{\top} \left(\left(\boldsymbol{\Sigma}_t^n + \boldsymbol{\mu}_t^n \boldsymbol{\mu}_t^{n\top} \right) - \gamma_k(\mathbf{Y}^n, t) \boldsymbol{\mu}_t^{n\top} - \boldsymbol{\mu}_t^n \gamma_k(\mathbf{Y}^n, t)^{\top} + \gamma_k(\mathbf{Y}^n, t) \gamma_k(\mathbf{Y}^n, t)^{\top} \right).\end{aligned}$$

B.1.3 Partition Function

Here we show how to compute the partition function $Z(\mathbf{Y})$. Note that this calculation is not used in the *Combining Discriminative Features* model, since it becomes computationally intractable once switching variables are introduced. Nonetheless, efficiently evaluating the partition function for the non-switching case is still an interesting problem.

To begin with, it helps to consider the easier case of computing the normalization constant of a product of unnormalized Gaussians:

$$f(\mathbf{x}) = \exp\left(-\frac{1}{2} \sum_i (\mathbf{x} - \boldsymbol{\mu}_i)^{\top} \boldsymbol{\tau}_i (\mathbf{x} - \boldsymbol{\mu}_i) \right).$$

By completing the square (see Appendix A.5), $f(\mathbf{x})$ can be rewritten as a single unnormalized Gaussian times an additional factor that is constant in $\mathbf{x}$:

$$f(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\tau}(\mathbf{x}-\boldsymbol{\mu})\right)\exp\left(-\frac{1}{2}\left(\sum_i \boldsymbol{\mu}_i{}^\top\boldsymbol{\tau}_i\boldsymbol{\mu}_i - \boldsymbol{\mu}^\top\boldsymbol{\tau}\boldsymbol{\mu}\right)\right),$$

where $\boldsymbol{\tau} = \sum_i \boldsymbol{\tau}_i$ and $\boldsymbol{\mu} = (\boldsymbol{\tau}^{-1})(\sum_i \boldsymbol{\tau}_i\boldsymbol{\mu}_i)$. Integration is now straightforward:

$$\int_{\mathbf{X}} f(\mathbf{x})d\mathbf{x} = (2\pi)^{\frac{d}{2}}|\boldsymbol{\tau}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\left(\sum_i \boldsymbol{\mu}_i{}^\top\boldsymbol{\tau}_i\boldsymbol{\mu}_i - \boldsymbol{\mu}^\top\boldsymbol{\tau}\boldsymbol{\mu}\right)\right).$$

Computing the partition function can be done analogously. After completing the square, $Z(\mathbf{Y})$ can be written as:

$$\begin{aligned} Z(\mathbf{Y}) &= \int_{\mathbf{X}} \exp\left(-\frac{1}{2}(\mathbf{X}-\boldsymbol{\mu})^\top\mathbf{P}(\mathbf{X}-\boldsymbol{\mu})\right)d\mathbf{X} \\ &\qquad \exp\left(-\frac{1}{2}\left(\sum_{t,k}\gamma_k(\mathbf{Y},t)^\top\boldsymbol{\beta}_k\gamma_k(\mathbf{Y},t) - \boldsymbol{\mu}^\top\mathbf{P}\boldsymbol{\mu}\right)\right) \\ &= (2\pi)^{\frac{DT}{2}}|\mathbf{P}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\left(\sum_{t,k}\gamma_k(\mathbf{Y},t)^\top\boldsymbol{\beta}_k\gamma_k(\mathbf{Y},t) - \boldsymbol{\mu}^\top\mathbf{P}\boldsymbol{\mu}\right)\right), \end{aligned}$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is a concatenation of the state variables, and the joint mean $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_T)$ is a concatenation of the marginal means. $\mathbf{P}$, the joint precision, is formed by summing the contributions to the precision from each of the factors in the energy function (as in the *Product of Gaussians* operation described in Appendix A). It takes the following block-tridiagonal form:

$$\begin{bmatrix} \mathbf{V}_0^{-1} + T\alpha T + \mathbf{B} & -\alpha T^\top & & & & \\ -\alpha T & T\alpha T + \mathbf{A} + \mathbf{B} & -\alpha T^\top & & & \\ & -\alpha T & T\alpha T + \mathbf{A} + \mathbf{B} & -\alpha T^\top & & \\ & & & \vdots & & \\ & & & -\alpha T & \mathbf{A} + \mathbf{B} \end{bmatrix}.$$

Due to the size of of $\mathbf{P}$, $DT \times DT$, this method for evaluating the partition function can be very expensive.

Another more efficient method uses the Markov property to factorize the conditional probability into $\mathrm{P}(\mathbf{X}|\mathbf{Y}) = \mathrm{P}(\mathbf{x}_1|\mathbf{Y})\mathrm{P}(\mathbf{x}_2|\mathbf{x}_1, \mathbf{Y})\ldots\mathrm{P}(\mathbf{x}_T|\mathbf{x}_{T-1}, \mathbf{Y})$. Each factor can be

expessed as $\mathrm{P}(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{Y}) = \mathrm{P}(\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{Y})/\mathrm{P}(\mathbf{x}_{t-1}|\mathbf{Y})$, which has a normalization constant of $(2\pi)^{\frac{D}{2}}|\mathbf{\Sigma}_{pair(t-1,t)}|^{\frac{1}{2}}|\mathbf{\Sigma}_{t-1}|^{-\frac{1}{2}}$. ($\mathbf{\Sigma}_t$ is the $D\times D$ marginal covariance and $\mathbf{\Sigma}_{pair(t-1,t)}$ the $2D\times 2D$ pairwise marginal covariance, which are computed during inference.) Thus the partition function can be evaluated as:

$$\log Z(\mathbf{Y}) = \frac{TD}{2}\log 2\pi + \frac{1}{2}\sum_{t=2}^{\top}\log|\mathbf{\Sigma}_{t-1,t}| - \frac{1}{2}\sum_{t=2}^{T-1}\log|\mathbf{\Sigma}_t| \\ -\frac{1}{2}\sum_{t,k}\gamma_k(\mathbf{Y},t)^{\top}\boldsymbol{\beta}_k\gamma_k(\mathbf{Y},t) + \frac{1}{2}\boldsymbol{\mu}^{\top}\mathbf{P}\boldsymbol{\mu}.$$

Furthermore, noting the sparse pattern of $\mathbf{P}$, we can can reexpress $\boldsymbol{\mu}^{\top}\mathbf{P}\boldsymbol{\mu}$ as:

$$\boldsymbol{\mu}_1{}^{\top}(T\alpha T+\mathbf{B})\boldsymbol{\mu}_1 + \sum_{t=2}^{T-1}\boldsymbol{\mu}_t{}^{\top}(T\alpha T+\mathbf{A}+\mathbf{B})\boldsymbol{\mu}_t + \boldsymbol{\mu}_T{}^{\top}(\mathbf{A}+\mathbf{B})\boldsymbol{\mu}_T - 2\sum_{t=2}^{\top}\boldsymbol{\mu}_t{}^{\top}\alpha T\boldsymbol{\mu}_{t-1}.$$

B.2 The Robust, Non-Gaussian Case

The previous approach used Gaussian distributions to model all sources of measurement error and noise. This choice, although permitting exact inference and gradient computations, restricts the distributions over the state to be unimodal at all time steps. The product of two Gaussians is a Gaussian, thus when we multiplicatively combine two disparate predictions, the result is a weighted average of the two. Instead, it makes more sense for the result to be a bimodal distribution. This can be achieved in a number of ways, such as by using product of 'robust' or 'heavy-tailed' error distributions like the "uniGauss" (an unnormalized mixture of a uniform and a Gaussian distribution, Hinton, 1999), or by using a mixture of Gaussians. The difficulty with the robust approach is that computing the partition function and gradient is no longer computationally feasible, requiring us to employ approximate Contrastive Divergence learning (Hinton, 2000).

The uniGauss model is obtained from the linear-Gaussian model by adding new unobserved binary variables, u and v, as illustrated in Figure B.3. Each binary variable is a switch which indicates whether or not the corresponding Gaussian feature should be

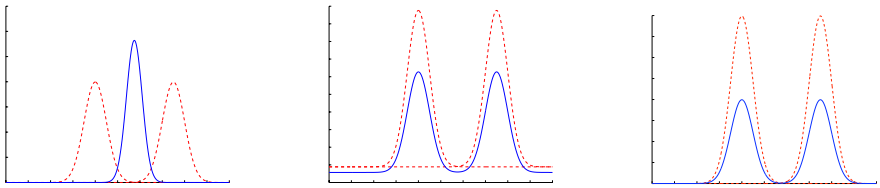

Figure B.2: The product of two Gaussians is another unimodal Gaussian (left), while the product of two robust uniGauss distributions with different means (middle) and a mixture of two Gaussians (right) are both bimodal.

included in the resulting joint distribution. Variable $u_{jt}^n = 1$ indicates that the j^{th} dynamics feature from $\mathbf{x}_{t-1}^n$ to $\mathbf{x}_t^n$ should be included, and $v_{kt}^n = 1$ that the k^{th} observation feature at time t should be included. Each observation switch is allowed to take on a value independently of the other switches, producing a product of uniGauss distributions (Hinton, 2000). In contrast, we constrain the dynamics switches so that exactly one is active at each time-step t: $\forall t \ \sum_j u_{jt} = 1$. This acts as a mixture of Gaussians, and ensures that even during occlusions, when all observation features are off, we still have a prediction for the state $\mathbf{x}_t$.

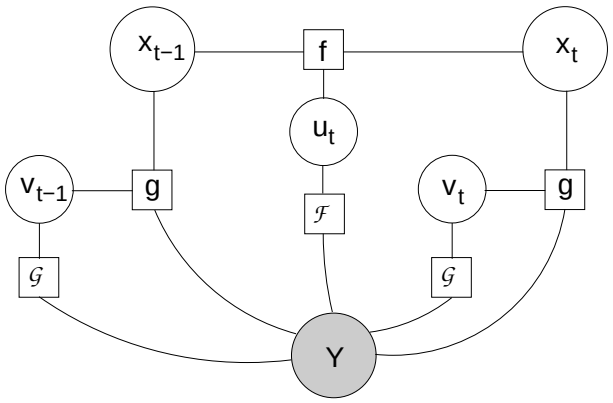

Figure B.3: The factor graph of the robust *Combining Discriminative Features* model.

Switching variables can themselves be conditioned on the values of various feature functions, called *switch potentials*. We introduce two additional sets of features: $\mathcal{F}_j(\mathbf{Y}^n, t)$,

which affect the probability of the dynamics switches u_{jt}^n, and $\mathcal{G}_k(\mathbf{Y}^n, t)$, which affect the observation switches v_{kt}^n. Each is a linear combination of a set of observation features and a learned parameter vector:

$$\mathcal{F}_j(\mathbf{Y}^n, t) = r_j(\mathbf{Y}^n, t)^\top \boldsymbol{\rho}_j$$
$$\mathcal{G}_k(\mathbf{Y}^n, t) = s_k(\mathbf{Y}^n, t)^\top \boldsymbol{\eta}_k.$$

The resulting log-probability, equivalent to (3.1), is

$$\mathcal{L} = \sum_{n=1}^{N} \left(\log \sum_{\mathbf{U},\mathbf{V}} \exp \left(\sum_j \operatorname{tr}(\mathbf{F}_j^n \boldsymbol{\alpha}_j) + \sum_k \operatorname{tr}(\mathbf{G}_k^n \boldsymbol{\beta}_k) + \sum_j \mathbf{R}_j^{n\top} \boldsymbol{\rho}_j + \sum_k \mathbf{S}_k^{n\top} \boldsymbol{\eta}_k \right) - \log Z(\mathbf{Y}^n) \right),$$

using the shorthand notation:

$$\mathbf{F}_j^n = -\frac{1}{2} \sum_{t=2}^{\top} \left(\mathbf{x}_t^n - \mathbf{T}_j \mathbf{x}_{t-1}^n\right) \left(\mathbf{x}_t^n - \mathbf{T}_j \mathbf{x}_{t-1}^n\right)^\top u_{jt}^n$$

$$\mathbf{G}_k^n = -\frac{1}{2} \sum_{t=1}^{\top} \left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right) \left(\mathbf{x}_t^n - \gamma_k(\mathbf{Y}^n, t)\right)^\top v_{kt}^n$$

$$\mathbf{R}_j^n = \sum_{t=2}^{\top} r_j(\mathbf{Y}^n, t) u_{jt}^n \qquad \mathbf{S}_k^n = \sum_{t=1}^{\top} s_k(\mathbf{Y}^n, t) v_{kt}^n.$$

The gradients are needed for learning parameters are:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_j} = \sum_{n=1}^{N} \left(\mathrm{E}_{\mathbf{U}|\mathbf{X}^n,\mathbf{Y}^n} \left[\mathbf{F}_j^n\right] - \mathrm{E}_{\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y}^n} \left[\mathbf{F}_j^n\right] \right)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}_k} = \sum_{n=1}^{N} \left(\mathrm{E}_{\mathbf{V}|\mathbf{X}^n,\mathbf{Y}^n} \left[\mathbf{G}_k^n\right] - \mathrm{E}_{\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y}^n} \left[\mathbf{G}_k^n\right] \right)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\rho}_j} = \sum_{n=1}^{N} \left(\mathrm{E}_{\mathbf{U}|\mathbf{X}^n,\mathbf{Y}^n} \left[\mathbf{R}_j^n\right] - \mathrm{E}_{\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y}^n} \left[\mathbf{R}_j^n\right] \right)$$

$$\frac{\partial \mathcal{L}}{\partial \eta_j} = \sum_{n=1}^{N} \left(\mathrm{E}_{\mathbf{V}|\mathbf{X}^n,\mathbf{Y}^n} \left[\mathbf{S}_k^n\right] - \mathrm{E}_{\mathbf{X},\mathbf{U},\mathbf{V}|\mathbf{Y}^n} \left[\mathbf{S}_k^n\right] \right).$$

Positive Phase

Here we need to compute the expectation over $\mathbf{U}$ and $\mathbf{V}$, given the fully observed training data $\mathbf{X}^n$ and $\mathbf{Y}^n$, of the features: $\mathrm{E}_{\mathbf{U},\mathbf{V}|\mathbf{X}^n,\mathbf{Y}^n}[feature]$. This expectation can be performed exactly, since the u's and v's are conditionally independent given $\mathbf{X}$ and $\mathbf{Y}$. All that needs to be done is to compute the probability (expected values) $\hat{u}^n_{jt}$ and $\hat{v}^n_{kt}$ of $\mathbf{U}$ and $\mathbf{V}$, and substitute them in place of u and v in the computation of the features $\mathbf{F}^n_j$, $\mathbf{G}^n_k$, $\mathbf{R}^n_j$, and $\mathbf{S}^n_k$.

$$\hat{u}^n_{jt} = \text{sigmoid}(f_j(\mathbf{x}^n_{t-1}, \mathbf{x}^n_t, \boldsymbol{\alpha}_j) + r_j(\mathbf{Y}^n, t)^\top \boldsymbol{\rho}_j)$$

$$\hat{v}^n_{kt} = \text{sigmoid}(g_k(\mathbf{x}^n_t, \mathbf{Y}^n, \boldsymbol{\beta}_k) + s_k(\mathbf{Y}^n, t)^\top \boldsymbol{\eta}_k).$$

Negative Phase

Here we have to compute the expectations of the features with respect to $\mathbf{X}, \mathbf{U}, \mathbf{V}|\mathbf{Y}^n$. In the robust model, this expectation cannot be computed exactly, and is computationally expensive to approximate. As described in Section 3.4 of Chapter 3, Contrastive Divergence is used to obtain biased samples of $\mathbf{X}, \mathbf{U}, \mathbf{V}$ given $\mathbf{Y}^n$.

Appendix C

Learning Articulated Structure From Motion: EM Algorithm

C.1 Derivation of Objective Function

Notation

D_o	dimensionality of observations (2 or 3)
D_l	dimensionality of local coordinate systems (3)
P	number of observed feature points
F	number of observed frames
J	number of vertices
S	number of sticks
i	indexes the endpoints, two for each stick
$s(i)$	the stick to which endpoint i belongs, $s(i) \equiv \lceil \frac{i}{2} \rceil$
$\Gamma()$	the Gamma function (Weisstein, b; Beal, 2003)
$\Psi()$	the Digamma function (Weisstein, a; Beal, 2003)

Joint Probability of Model:

$$\begin{aligned}\mathbb{P} = \mathrm{P}(\mathbf{W}|\mathbf{M},\mathbf{L},\mathbf{R})\ \mathrm{P}(\mathbf{E}|\mathbf{M},\mathbf{K},\mathbf{V},\boldsymbol{\phi},\mathbf{G}) \\ \mathrm{P}(\mathbf{V})\ \mathrm{P}(\boldsymbol{\phi})\ \mathrm{P}(\mathbf{M})\ \mathrm{P}(\mathbf{L})\ \mathrm{P}(\mathbf{K})\ \mathrm{P}(\mathbf{R})\ \mathrm{P}(\mathbf{G})\end{aligned}$$

Variational Posterior

$$\begin{aligned}\mathbb{Q} &= Q(\mathbf{V})\ Q(\mathbf{E})\ Q(\phi) \\ Q(\mathbf{V}) &= \prod_{f,j} \mathcal{N}(\mathbf{v}_j^f|\mu(\mathbf{v}_j^f),\tau(\mathbf{v}_j^f)) \\ Q(\mathbf{E}) &= \prod_{f,i} \mathcal{N}(\mathbf{e}_i^f|\mu(\mathbf{e}_i^f),\tau(\mathbf{e}_i^f)) \\ Q(\boldsymbol{\phi}) &= \prod_{j} \mathrm{Gamma}(\phi_j|\alpha(\phi_j),\beta(\phi_j))\end{aligned}$$

Expected Complete Log-Probability (Negative Free Energy):

EM learning of the model parameters seeks to maximize the following objective function:

$$\mathcal{L} = \mathrm{E}_{\mathbb{Q}}[\log \mathbb{P}] - \mathrm{E}_{\mathbb{Q}}[\log \mathbb{Q}]. \tag{C.1}$$

We now derive each of the individual terms appearing in the objective function, then bring them together, presenting $\mathcal{L}$ in full detail (equation C.2). When evaluating the following expectations, we make use of the *Expectation of a Quadratic* identity, introduced in Appendix A.4, and the formula for the expectation of the log of a Gamma random variable (Beal, 2003).

Observation Likelihood

$$\begin{aligned}&\mathrm{E}_{\mathbb{Q}}\left[\log \mathrm{P}(\mathbf{W}|\mathbf{M},\mathbf{L},\mathbf{R})\right] \\ &= \mathrm{E}_{\mathbb{Q}}\left[\sum_{f,p,s} r_{p,s}\left(-\frac{\tau_w}{2}\|\mathbf{w}_p^f - \mathbf{M}_s^f\mathbf{l}_{s,p}\|^2 + \frac{D_o}{2}\log(\tau_w) - \frac{D_o}{2}\log(2\pi)\right)\right] \\ &= -\frac{\tau_w}{2}\sum_{f,p,s}\|\mathbf{w}_p^f - \mathbf{M}_s^f\mathbf{l}_{s,p}\|^2 + \frac{FPD_o}{2}\log(\tau_w) - \frac{FPD_o}{2}\log(2\pi)\end{aligned}$$

Pseudo-Observation Likelihood

The pseudo-observation likelihood, $\mathrm{P}(\mathbf{E}|\mathbf{M},\mathbf{K},\mathbf{M},\boldsymbol{\phi},\mathbf{G})$ consists of two factors.

$$\begin{aligned}
&\mathrm{E}_{\mathbb{Q}}\left[\log\prod_{f,i}\mathcal{N}(\mathbf{e}_i^f|\mathbf{M}_{s(i)}^f\mathbf{k}_i,\tau_m^{-1}\mathbf{I})\right]\\
&\quad=\mathrm{E}_{\mathbb{Q}}\left[\sum_{f,i}\left(-\frac{\tau_m}{2}\|\mathbf{e}_i^f-\mathbf{M}_{s(i)}^f\mathbf{k}_i\|^2+\frac{D_l}{2}\log(\tau_m)-\frac{D_l}{2}\log(2\pi)\right)\right]\\
&\quad=-\frac{\tau_m}{2}\sum_{f,i}\left(\|\mu(\mathbf{e}_i^f)-\mathbf{M}_{s(i)}^f\mathbf{k}_i\|^2+D_l\tau(e_i^f)^{-1}\right)+FSD_l\log(\tau_m)-FSD_l\log(2\pi)\\
&\mathrm{E}_{\mathbb{Q}}\left[\log\prod_{f,i,j}\mathcal{N}(\mathbf{e}_i^f|\mathbf{v}_j^f,\phi_j^{-1}\mathbf{I})^{g_{i,j}}\right]\\
&\quad=\mathrm{E}_{\mathbb{Q}}\left[\sum_{f,i,j}g_{i,j}\left(-\frac{\phi_j}{2}\|\mathbf{e}_i^f-\mathbf{v}_j^f\|^2+\frac{D_l}{2}\log(\phi_j)-\frac{D_l}{2}\log(2\pi)\right)\right]\\
&\quad=-\frac{1}{2}\sum_{f,i,j}g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)}\left(\|\mu(\mathbf{e}_i^f)-\mu(\mathbf{v}_j^f)\|^2+D_l(\tau(\mathbf{e}_i^f)^{-1}+\tau(\mathbf{v}_j^f)^{-1})\right)\\
&\qquad+\frac{FD_l}{2}\sum_{i,j}g_{i,j}\left(\Psi(\alpha(\phi_j))-\log\beta(\phi_j)\right)-FSD_l\log(2\pi)
\end{aligned}$$

Smoothing Factor

$$\begin{aligned}
\mathrm{E}_{\mathbb{Q}}\left[\log\mathrm{P}(\mathbf{V})\right]&=\mathrm{E}_{\mathbb{Q}}\left[\sum_j\sum_{f=2}^{F}\left(-\frac{\tau_t}{2}\|\mathbf{v}_j^f-\mathbf{v}_j^{f-1}\|^2-\frac{D_l}{2}\log(2\pi)+\frac{D_l}{2}\log(\tau_t)\right)\right]\\
&=-\frac{\tau_t}{2}\sum_j\sum_{f=2}^{F}\left(\|\mathbf{v}_j^f-\mathbf{v}_j^{f-1}\|^2+D_l(\tau(\mathbf{v}_j^{f-1})^{-1}+\tau(\mathbf{v}_j^f)^{-1})\right)\\
&\quad-\frac{(F-1)JD_l}{2}\log(2\pi)+\frac{(F-1)JD_l}{2}\log(\tau_t)
\end{aligned}$$

Joint Precisions

$$\begin{aligned}
\mathrm{E}_{\mathbb{Q}}\left[\log\mathrm{P}(\boldsymbol{\phi})\right]&=\mathrm{E}_{\mathbb{Q}}\left[\log\prod_j\mathrm{Gamma}(\phi_j|\alpha_j,\beta_j)\right]\\
&=\mathrm{E}_{\mathbb{Q}}\left[\sum_j(\alpha_j\log\beta_j+(\alpha_j-1)\log\phi_j-\beta_j\phi_j-\log\Gamma(\alpha_j)\right]\\
&=-\sum_j\beta_j\frac{\alpha(\phi_j)}{\beta(\phi_j)}+\sum_j(\alpha_j-1)(\Psi(\alpha(\phi_j))-\log\beta(\phi_j)))+\sum_j\alpha_j\log\beta_j\sum_j\log\Gamma(\alpha_j)
\end{aligned}$$

Structure Priors

$$\mathrm{E}_{\mathbb{Q}}\left[\log \mathrm{P}(\mathbf{R})\right] = \sum_{p,s} r_{p,s} \log c_S \qquad \mathrm{E}_{\mathbb{Q}}\left[\log \mathrm{P}(\mathbf{G})\right] = \sum_{i,j} g_{i,j} \log c_j$$

Entropy Terms

Entropy of $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \tau^{-1}\mathbf{I}_{D\times D})$: $H(x) = -D/2 \log \tau + D/2 \log(2\pi e)$

Entropy of Gamma$(\phi|\alpha, \beta)$: $H(\phi) = \alpha - \log \beta + \log \Gamma(\alpha) + (1-\alpha)\Psi(\alpha)$

$$\mathrm{E}_{\mathbb{Q}}\left[\log Q(\mathbf{V})\right] = -\frac{D_l}{2} \sum_{f,j} \log \tau(\mathbf{v}_j^f) + \frac{FJD_l}{2} \log(2\pi e)$$

$$\mathrm{E}_{\mathbb{Q}}\left[\log Q(\mathbf{E})\right] = -\frac{D_l}{2} \sum_{f,i} \log \tau(\mathbf{e}_i^f) + FSD_l \log(2\pi e)$$

$$\mathrm{E}_{\mathbb{Q}}\left[\log Q(\boldsymbol{\phi})\right] = \sum_j (\alpha(\phi_j) - \log \beta(\phi_j) + \log \Gamma(\alpha(\phi_j)) + (1-\alpha(\phi_j))\Psi(\alpha(\phi_j)))$$

Adding the above terms together, we arrive at the following expression for the

Expected Complete Log-Likelihood objective function, (C.1):

$$
\begin{aligned}
\mathcal{L} = & -\frac{\tau_w}{2}\sum_{f,p,s} r_{p,s}\|\mathbf{w}_p^f - \mathbf{M}_s^f\mathbf{l}_{s,p}\|^2 + \frac{FPD_o}{2}\log\tau_w - \frac{FPD_o}{2}\log(2\pi) \\
& -\frac{1}{2}\sum_{f,i,j} g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)}\big(\|\mu(\mathbf{e}_i^f) - \mu(\mathbf{v}_j^f)\|^2 - D_o(\tau(\mathbf{e}_i^f)^{-1} + \tau(\mathbf{v}_j^f)^{-1})\big) \\
& \quad + \frac{FD_o}{2}\sum_{i,j}(\Psi(\alpha(\phi_j)) - \log\beta(\phi_j)) - FSD_o\log(2\pi) \\
& -\frac{\tau_m}{2}\sum_{f,i}\|\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f\mathbf{k}_i\|^2 - \frac{Do}{2}\tau_m\sum_{f,i}\tau(\mathbf{e}_i^f)^{-1} + FSD_o\log(\tau_m) - FSD_o\log(2\pi) \\
& -\frac{\tau_t}{2}\sum_{f=2}^{F}\sum_j\|\mu(\mathbf{v}_j^f) - \mu(\mathbf{v}_j^{f-1})\|^2 - \frac{D_o}{2}\tau_t\sum_{f,j}\tau(\mathbf{v}_j^f)^{-1}2^{h(f)} + \frac{(F-1)JD_o}{2}\log(\tau_t) \\
& \quad - \frac{(F-1)JD_o}{2}\log(2\pi) \\
& -\sum_j\beta_j\frac{\alpha(\phi_j)}{\beta(\phi_j)} + \sum_j(\alpha_j - 1)(\Psi(\alpha(\phi_j)) - \log\beta(\phi_j)) + \sum_j\alpha_j\log\beta_j - \sum_j\log\Gamma(\alpha_j) \\
& -\frac{D_o}{2}\sum_{f,j}\log\tau(\mathbf{v}_j^f) + \frac{FJD_o}{2}\log(2\pi e) - \frac{D_o}{2}\sum_{f,i}\log\tau(\mathbf{e}_i^f) + FSD_o\log(2\pi e) \\
& +\sum_j\alpha(\phi_j) - \sum_j\log\beta_j + \sum_j\log\Gamma(\alpha(\phi_j)) + \sum_j(1-\alpha(\phi_j))\Psi(\alpha(\phi_j)) \\
& -\frac{\tau_p}{2}\sum_{s,p} r_{s,p}\|\mathbf{l}_{s,p}\|^2 + \frac{PD_o}{2}\log\tau_p - \frac{\tau_p}{2}\sum_i\|\mathbf{k}_i\|^2 + SD_o\log\tau_p
\end{aligned}
\tag{C.2}
$$

where $h(f) = 1$ if $1 < f < F$ and 0 otherwise, and $s(i)$ is simply the index of the stick to which endpoint i belongs.

C.2 EM Updates

The EM updates for model are obtained by maximizing the objective function (C.2) with respect to each parameter. Included below are the gradients of $\mathcal{L}$ (which can be obtained using the formulas in Petersen and Pedersen, 2008) and the resulting parameter updates. Also included, for completeness, are EM updates for the structure variables, $\mathbf{R}$ and $\mathbf{G}$.

Update of τ_w

$$\frac{\partial \mathcal{L}}{\partial \tau_w^{-1}} = -\frac{1}{2}\sum_{f,p,s} r_{p,s}\|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2 + \frac{FPD_o}{2}\tau_w^{-1}$$
$$\tau_w^{-1} = \frac{\sum_{f,p,s} r_{p,s}\|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2}{FPD_o}$$

Update of τ_m

$$\frac{\partial \mathcal{L}}{\partial \tau_m^{-1}} = -\frac{1}{2}\sum_{f,i} \|\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f \mathbf{k}_i\|^2 - \frac{D_o}{2}\sum_{f,i}\tau(\mathbf{e}_i^f)^{-1} + FSD_o\tau_m^{-1}$$
$$\tau_m^{-1} = \frac{\sum_{f,i} \|\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f \mathbf{k}_i\|^2}{2FSD_o} + \frac{\sum_{f,i}\tau(\mathbf{e}_i^f)^{-1}}{2FS}$$

Update of τ_w if we define $\tau_m \equiv \tau_w$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tau_w^{-1}} =& -\frac{1}{2}\sum_{f,p,s} r_{p,s}\|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2 + \frac{FPD_o}{2}\tau_w^{-1} \\ & -\frac{1}{2}\sum_{f,i} \|\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f \mathbf{k}_i\|^2 - \frac{D_o}{2}\sum_{f,i}\tau(\mathbf{e}_i^f)^{-1} + FSD_o\tau_m^{-1} \\ \tau_w^{-1} =& \frac{\sum_{f,p,s} r_{p,s}\|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2 + \sum_{f,i} \|\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f \mathbf{k}_i\|^2 + D_o\sum_{f,i}\tau(\mathbf{e}_i^f)^{-1}}{F(P+2S)D_o}\end{aligned}$$

Update of τ_t

$$\frac{\partial \mathcal{L}}{\partial \tau_t^{-1}} = -\frac{1}{2}\sum_{f=2}^{F}\sum_{j} \|\mu(\mathbf{v}_j^f) - \mu(\mathbf{v}_j^{f-1})\|^2 - \frac{D_o}{2}\sum_{f,j}\tau(\mathbf{v}_j^f)2^{h(f)} + \frac{(F-1)JD_o}{2}\tau_t^{-1}$$
$$\tau_t^{-1} = \frac{\sum_{f=2}^{F}\sum_j \|\mu(\mathbf{v}_j^f) - \mu(\mathbf{v}_j^{f-1})\|^2}{(F-1)JD_o} + \frac{\sum_{f,j}\tau(\mathbf{v}_j^f)^{-1}}{(F-1)J}2^{h(f)}$$

Update of $\mathbf{M}_s^f$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{M}_s^f} = -\frac{\tau_w}{2}\sum_p r_{p,s}\Big(-2\mathbf{w}_p^f \mathbf{l}_{s,p}{}^\top + 2\mathbf{M}_s^f \mathbf{l}_{s,p}\mathbf{l}_{s,p}{}^\top\Big) - \frac{\tau_m}{2}\sum_{\{i|s(i)=s\}}\Big(-2\mu(\mathbf{e}_i^f)\mathbf{k}_{s,i}{}^\top + 2\mathbf{M}_s^f \mathbf{k}_{s,i}\mathbf{k}_{s,i}{}^\top\Big)$$
$$\mathbf{M}_s^f = \Big(\tau_w\sum_f r_{p,s}\mathbf{l}_{s,p}\mathbf{l}_{s,p}{}^\top + \tau_m\sum_{\{i|s(i)=s\}}\mathbf{k}_{s(i)}\mathbf{k}_{s(i)}{}^\top\Big)^{-1}\Big(\tau_w\sum_p r_{p,s}\mathbf{w}_p^f \mathbf{l}_{s,p}{}^\top + \tau_m\sum_{\{i|s(i)=s\}}\mu(\mathbf{e}_i^f)\mathbf{k}_{s(i)}{}^\top\Big)$$

However this doesn't enforce orthogonality of the rotation part of $\mathbf{M}_s^f$. Instead split the motion matrix $\mathbf{M}_s^f$ into a rotation matrix and a translation vector $\mathbf{M}_s^f = [\mathbf{R}_s^f \quad \mathbf{t}_{s,f}]$, and solve for each of them independently.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{t}_{s,f}} = -\frac{\tau_w}{2}\sum_p r_{p,s}\Big(2\mathbf{t}_{s,f} - 2(\mathbf{w}_p^f - \mathbf{R}_s^f \mathbf{l}_{s,p})\Big) - \frac{\tau_m}{2}\sum_{\{i|s(i)=s\}}\Big(2\mathbf{t}_{s(i)}^f - 2(\mu(\mathbf{e}_i^f) - \mathbf{M}_s^f \mathbf{k}_{s,i})\Big)$$

$$\mathbf{t}_{s,f} = \Big(\tau_w \sum_p r_{p,s}(\mathbf{w}_p^f - \mathbf{R}_s^f \mathbf{l}_{s,p}) + \tau_m \sum_{\{i|s(i)=s\}}(\mu(\mathbf{e}_i^f) - \mathbf{M}_s^f \mathbf{k}_{s,i})\Big) / \Big(\tau_w \sum_p r_{p,s} + 2\tau_m\Big)$$

Another reasonable update for $\mathbf{t}_{s,f}$ is:

$$\mathbf{t}_{s,f} = \Big(\sqrt{\tau_w}\sum_p r_{p,s}\mathbf{w}_p^f + \sqrt{\tau_m}\sum_{\{i|s(i)=s\}}\mu(\mathbf{e}_i^f)\Big) / \Big(\sqrt{\tau_w}\sum_p r_{p,s} + 2\sqrt{\tau_m}\Big)$$

To solve for $\mathbf{R}_s^f$ we can rewrite it as an orthogonal Procrustes problem (Golub and Van Loan, 1996; Viklands, 2006). First collect all the terms of $\mathcal{L}$ that involve rotation:

$$\mathcal{L} = -\frac{\tau_w}{2}\sum_{f,p,s} r_{p,s}\|(\mathbf{w}_p^f - \mathbf{t}_{s,f}) - \mathbf{R}_s^f \mathbf{l}_{s,p}\|^2 - \frac{\tau_m}{2}\sum_{f,i}\|(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s,f}) - \mathbf{R}_s^f \mathbf{k}_{s,i}\|^2.$$

Now to obtain the optimal $\mathbf{R}_s^f$, we must the problem $\text{argmin}_{\mathbf{R}_s^f}\|\mathbf{A} - \mathbf{R}_s^f\mathbf{B}\|^2$, where

$$\mathbf{A} = \Big[\big[\sqrt{\tau_w}\, r_{p,s}(\mathbf{w}_p^f - \mathbf{t}_{s,f})\big]_{p=1..P} \quad \big[\sqrt{\tau_m}\,(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s,f})\big]_{\{i|s(i)=s\}}\Big]$$

$$\mathbf{B} = \Big[\big[\sqrt{\tau_w}\, r_{p,s}\mathbf{l}_{s,p}\big]_{p=1..P} \quad \big[\sqrt{\tau_m}\,\mathbf{k}_i\big]_{\{i|s(i)=s\}}\Big].$$

Update of $\mathbf{l}_{s,p}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{l}_{s,p}} = -\frac{\tau_w}{2}\sum_f(-2\mathbf{M}_s^{f\top}\mathbf{w}_p^f + 2\mathbf{M}_s^{f\top}\mathbf{M}_s^f\mathbf{l}_{s,p}) - \tau_p\mathbf{l}_{s,p}$$

$$\mathbf{l}_{s,p} = \big(\sum_f \mathbf{M}_s^{f\top}\mathbf{M}_s^f + \frac{\tau_p}{\tau_w}\mathbf{I}\big)^{-1}\sum_f \mathbf{M}_s^{f\top}\mathbf{w}_p^f$$

However, the above update doesn't require that the last coordinate of $\mathbf{l}_{s,p}$, a homogeneous vector, must always be one. To take care of this the relevant terms of the expected logprob can be rewritten as:

$$\mathcal{L} = -\frac{1}{2}\tau_w\sum_{f,p,s} r_{p,s}\left\|\mathbf{w}_p^f - \begin{bmatrix}\mathbf{R}_s^f & \mathbf{t}_{s,f}\end{bmatrix}\begin{bmatrix}\mathbf{l}_{s,p}\\ 1\end{bmatrix}\right\|^2 - \frac{1}{2}\tau_p\sum_{s,p} r_{s,p}\|\mathbf{l}_{s,p}\|^2 + \text{terms constant wrt } \mathbf{l}_{s,p}$$

where $\mathbf{l}_{s,p}$ is now a 3-vector, and $\mathbf{M}_s^f$ is separated into its rotation component $\mathbf{R}_s^f$ and translation vector $\mathbf{t}_{s,f}$. The resulting update becomes:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{l}_{s,p}} = -\frac{\tau_w}{2}\sum_f(-2\mathbf{R}_s^{f\top}(\mathbf{w}_p^f - \mathbf{t}_{s,f}) + 2\mathbf{R}_s^{f\top}\mathbf{R}_s^f\mathbf{l}_{s,p}) - \tau_p\mathbf{l}_{s,p}$$
$$\mathbf{l}_{s,p} = \big(\sum_f \mathbf{R}_s^{f\top}\mathbf{R}_s^f + \frac{\tau_p}{\tau_w}\mathbf{I}\big)^{-1}\sum_f \mathbf{R}_s^{f\top}(\mathbf{w}_p^f - \mathbf{t}_{s,f})$$

Update of $\mathbf{k}_i$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{k}_i} = -\frac{\tau_m}{2}\sum_f 2\big(\mathbf{M}_{s(i)}^{f\top}\mathbf{M}_{s(i)}^f\mathbf{k}_i - \mathbf{M}_{s(i)}^{f\top}\mu(\mathbf{e}_i^f)\big) - \tau_p\mathbf{k}_i$$
$$\mathbf{k}_i = \big(\sum_f \mathbf{M}_{s(i)}^{f\top}\mathbf{M}_{s(i)}^f + \frac{\tau_p}{\tau_m}\mathbf{I}\big)^{-1}\sum_f \mathbf{M}_{s(i)}^{f\top}\mu(\mathbf{e}_i^f)$$

As with $\mathbf{l}_{s,p}$, when homogeneity of $\mathbf{k}_i$ is enforced, the update becomes

$$\mathbf{k}_i = \big(\sum_f \mathbf{R}_{s(i)}^{f\top}\mathbf{R}_{s(i)}^f + \frac{\tau_p}{\tau_m}\mathbf{I}\big)^{-1}\sum_f \mathbf{R}_{s(i)}^{f\top}(\mu(\mathbf{e}_i^f) - \mathbf{t}_{s(i)}^f)$$

Update of $\mu(\mathbf{e}_i^f)$

$$\frac{\partial \mathcal{L}}{\partial \mu(\mathbf{e}_i^f)} = -\frac{1}{2}\sum_j g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)}2(\mu(\mathbf{e}_i^f) - \mu(\mathbf{v}_j^f)) - \frac{1}{2}\tau_m 2(\mu(\mathbf{e}_i^f) - \mathbf{M}_{s(i)}^f\mathbf{k}_i)$$
$$\mu(\mathbf{e}_i^f) = \frac{\tau_m\mathbf{M}_{s(i)}^f\mathbf{k}_i + \sum_j g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)}\mu(\mathbf{v}_j^f)}{\tau_m + \sum_j g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)}}$$

Update of $\tau(\mathbf{e}_i^f)$

$$\frac{\partial \mathcal{L}}{\partial \tau(\mathbf{e}_i^f)^{-1}} = -\frac{D_o}{2}\sum_j g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)} - \frac{D_o}{2}\tau_m + \frac{D_o}{2}\tau(\mathbf{e}_i^f)^{-1}$$
$$\tau(\mathbf{e}_i^f) = \sum_j g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)} + \tau_m$$

Update of $\mu(\mathbf{v}_j^f)$

$$\frac{\partial \mathcal{L}}{\partial \mu(\mathbf{v}_j^f)} = -\frac{1}{2}\sum_i g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)}(2\mu(\mathbf{v}_j^f) - 2\mu(\mathbf{e}_i^f)) - [f > 1]\frac{\tau_t}{2}(2\mu(\mathbf{v}_j^f) - 2\mu(\mathbf{v}_j^{f-1})) - [f < F]\frac{\tau_t}{2}(2\mu(\mathbf{v}_j^f) - 2\mu(\mathbf{v}_j^{f+1}))$$

$$\mu(\mathbf{v}_j^f) = \frac{\frac{\alpha(\phi_j)}{\beta(\phi_j)}\sum_i g_{i,j}\mu(\mathbf{e}_i^f) + [f > 1]\tau_t\mu(\mathbf{v}_j^{f-1}) + [f < F]\tau_t\mu(\mathbf{v}_j^{f+1})}{\frac{\alpha(\phi_j)}{\beta(\phi_j)}\sum_i g_{i,j} + \tau_t 2^{h(f)}}$$

Update of $\tau(\mathbf{v}_j^f)$

$$\frac{\partial \mathcal{L}}{\partial \tau(\mathbf{v}_j^f)^{-1}} = -\frac{D_o}{2}\sum_i g_{i,j}\frac{\alpha(\phi_j)}{\beta(\phi_j)} - \frac{D_o}{2}\tau_t 2^{h(f)} + \frac{D_o}{2}\tau(\mathbf{v}_j^f)^{-1}$$

$$\tau(\mathbf{v}_j^f) = \frac{\alpha(\phi_j)}{\beta(\phi_j)}\sum_i g_{i,j} + \tau_t 2^{h(f)}$$

Joint Precisions

Each joint has a precision $\phi_j \sim \text{Gamma}(\alpha_j, \beta_j)$. The Gamma prior and likelihood are conjugate, thus the posterior distribution over ϕ_j (conditioning on all data/variables) is also Gamma (Gelman et al., 2003), with parameters

$$\alpha(\phi_j) = \alpha_j + \frac{FD_o}{2}\sum_i g_{i,j}$$

$$\beta(\phi_j) = \beta_j + \frac{1}{2}\sum_{f,i} g_{i,j}\|\mu(\mathbf{e}_i^f) - \mu(\mathbf{v}_j^f)\|^2 + \frac{D_o}{2}\sum_{f,i} g_{i,j}[(\tau(\mathbf{e}_i^f))^{-1} + (\tau(\mathbf{v}_j^f))^{-1}]$$

The M-step updates of the α_j and β_j parameters are simply:

$$\alpha_j = \alpha(\phi_j) \qquad\qquad \beta_j = \beta(\phi_j)$$

Update of $c_{p,s}$

To enforce the requirement that $\forall p \ \sum_s c_{p,s} = 1$, we must include the terms $\sum_p \lambda_p^c(\sum_s c_{p,s} - 1)$ in the expected log-probability, where λ_p^c are Lagrange Multipliers.

$$\frac{\partial \mathcal{L}}{\partial c_{p,s}} = -\frac{r_{p,s}}{c_{p,s}} + \lambda_p^c$$
$$c_{p,s} = \frac{r_{p,s}}{\sum_{s'} r_{p,s'}}$$

Update of $d_{i,j}$

To enforce the requirement that $\forall i \ \sum_j d_{i,j} = 1$, we must include the term $\sum_i \lambda_i^d(\sum_j d_{i,j} - 1)$ in the expected log-probability, where λ_i^d are Lagrange Multipliers.

$$\frac{\partial \mathcal{L}}{\partial d_{i,j}} = \frac{g_{i,j}}{d_{i,j}} + \lambda_i^d$$
$$d_{i,j} = \frac{g_{i,j}}{\sum_{j'} g_{i,j'}}$$

Update of $r_{p,s}$

To enforce the requirement that $\forall p \ \sum_s r_{p,s} = 1$, we must include the terms $\sum_p \lambda_p^r(\sum_s r_{p,s} - 1)$ in the expected log-probability, where λ_p^r are Lagrange Multipliers.

$$\frac{\partial \mathcal{L}}{\partial r_{p,s}} = -\frac{1}{2}\tau_w \sum_f \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2 + \log c_{p,s} - \log r_{p,s} - 1 + \lambda_p^r$$
$$\mathrm{E}[r_{p,s}] \propto c_{p,s} \exp\left(-\frac{\tau_w}{2} \sum_f \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2\right) \quad \text{s.t.} \sum_{s'} r_{p,s'} = 1$$

Update of $g_{i,j}$

To enforce the requirement that $\forall i \ \sum_j g_{i,j} = 1$, we must include the term $\sum_i \lambda_i^g (\sum_j g_{i,j} - 1)$ in the expected log-probability, where λ_i^g are Lagrange Multipliers.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial g_{i,j}} =& -\frac{1}{2}\frac{\alpha(\phi_j)}{\beta(\phi_j)} \sum_f \left(\|\mu(\mathbf{e}_i^f) - \mu(\mathbf{v}_j^f)\|^2 - D_o(\tau(\mathbf{e}_i^f)^{-1} + \tau(\mathbf{v}_j^f)^{-1}) \right) \\ &+ \frac{FD_o}{2}(\Psi(\alpha(\phi_j)) - \log \beta(\phi_j)) + \log d_{i,j} - 1 - \log g_{i,j} + \lambda_i^g \\ \mathrm{E}[g_{i,j}] \propto& \exp\left(-\frac{1}{2}\frac{\alpha(\phi_j)}{\beta(\phi_j)} \sum_f \left(\|\mu(\mathbf{e}_i^f) - \mu(\mathbf{v}_j^f)\|^2 - D_o(\tau(\mathbf{e}_i^f)^{-1} + \tau(\mathbf{v}_j^f)^{-1}) \right) \right) \\ &\times \exp\left(\frac{FD_o}{2}\Psi(\alpha(\phi_j)) \right) \beta(\phi_j)^{-FD_o/2} \, d_{i,j} \quad \text{s.t.} \sum_{j'} g_{i,j'} = 1\end{aligned}$$

C.3 Accounting for Missing Data

To account for missing observations, we introduce a set of binary mask variables $m_{f,p}$, which indicate with a 1 if marker p is visible in frame f, or a 0 if it is not. Including the mask variables change the observation likelihood terms of the objective function (C.2) from

$$-\frac{1}{2}\tau_w \sum_{f,p,s} r_{p,s} \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2 + \frac{FPD_o}{2}\log \tau_w - \frac{FPD_o}{2}\log(2\pi)$$

to

$$-\frac{1}{2}\tau_w \sum_{f,p,s} m_{f,p} r_{p,s} \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2 + \frac{(\sum_{f,p} m_{f,p}) D_o}{2}\log \tau_w - \frac{(\sum_{f,p} m_{f,p}) D_o}{2}\log(2\pi).$$

This results in four minor changes to the EM updates, as detailed below.

Update of τ_w

$$\tau_w^{-1} = \frac{\sum_{f,p,s} m_{f,p} r_{p,s} \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2}{(\sum_{f,p} m_{f,p}) D_o}$$

Update of $\mathbf{M}_s^f$

$$\begin{aligned}\mathbf{M}_s^f =&\Big(\tau_w \sum_f m_{f,p} r_{p,s} \mathbf{l}_{s,p} \mathbf{l}_{s,p}{}^\top + \tau_m \sum_{\{i|s(i)=s\}} \mathbf{k}_{s(i)} \mathbf{k}_{s(i)}{}^\top\Big)^{-1} \\ &\times \Big(\tau_w \sum_p m_{f,p} r_{p,s} \mathbf{w}_p^f \mathbf{l}_{s,p}{}^\top + \tau_m \sum_{\{i|s(i)=s\}} \mu(\mathbf{e}_i^f) \mathbf{k}_{s(i)}{}^\top\Big)\end{aligned}$$

The alternative updates for $\mathbf{R}_s^f$ and $\mathbf{t}_s^f$ are

$$\begin{aligned}\mathbf{t}_{s,f} =&\Big(\tau_w \sum_p m_{f,p}\, r_{p,s} (\mathbf{w}_p^f - \mathbf{R}_s^f \mathbf{l}_{s,p}) + \tau_m \sum_{\{i|s(i)=s\}} (\mu(\mathbf{e}_i^f) - \mathbf{M}_s^f \mathbf{k}_{s,i})\Big)^{-1} \\ &\times \Big(\tau_w \sum_p m_{f,p}\, r_{p,s} + 2\tau_m\Big)\end{aligned}$$

and again solving the orthogonal Procrustes problem $\operatorname{argmin}_{\mathbf{R}_s^f} \|\mathbf{A} - \mathbf{R}_s^f \mathbf{B}\|^2$, where

$$\begin{aligned}\mathbf{A} &= \Big[\big[\sqrt{\tau_w}\, m_{f,p}\, r_{p,s} (\mathbf{w}_p^f - \mathbf{t}_{s,f})\big]_{p=1..P} \quad \big[\sqrt{\tau_m}\, (\mu(\mathbf{e}_i^f) - \mathbf{t}_{s,f})\big]_{\{i|s(i)=s\}}\Big] \\ \mathbf{B} &= \Big[\big[\sqrt{\tau_w}\, m_{f,p}\, r_{p,s}\, \mathbf{l}_{s,p}\big]_{p=1..P} \quad \big[\sqrt{\tau_m}\, \mathbf{k}_i\big]_{\{i|s(i)=s\}}\Big].\end{aligned}$$

Update of $\mathbf{l}_{s,p}$

$$\mathbf{l}_{s,p} = \left(\sum_f m_{f,p}\, \mathbf{R}_s^{f\top} \mathbf{R}_s^f + \frac{\tau_p}{\tau_w} \mathbf{I}\right)^{-1} \sum_f m_{f,p}\, \mathbf{R}_s^{f\top} (\mathbf{w}_p^f - \mathbf{t}_{s,f})$$

Update of $r_{p,s}$

$$r_{p,s} \propto c_{p,s} \exp\left(-\frac{\tau_w}{2} \sum_f m_{f,p} \|\mathbf{w}_p^f - \mathbf{M}_s^f \mathbf{l}_{s,p}\|^2\right) \quad \text{s.t.} \sum_{s'} r_{p,s'} = 1$$

Bibliography

K. Abdel-Malek, J. Arora, S. Beck, M. Bhatti, J. Carroll, T. Cook, S. Dasgupta, N. Grosland, R. Han, H. Kim, J. Lu, C. Swan, A. Williams, and J. Yang. Digital human modeling and virtual reality for FCS. Technical Report VSR-04.02, The Virtual Soldier Research (VSR) Program, Center for Computer-Aided Design, College of Engineering, The University of Iowa, October 2004.

Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, pages 1–20. MIT Press, 1991.

Shai Avidan. Support vector tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 184–191, 2001.

Matthew J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, The Gatsby Computational Neuroscience Unit, University College London, 2003.

Peter Belhumeur and D. Kreigman. What is the set of images of an object under all possible lighting conditions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 270–277, 1997.

Stan Birchfield. Elliptical head tracking using intensity gradient and color histograms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 232–37, 1998.

Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using view-based representation. In B. Buxton and R. Cipolla, editors, *Proceedings of the Fourth European Conference on Computer Vision (ECCV)*, LNCS 1064, pages 329–342. Springer Verlag, 1996.

Michael J. Black, David J. Fleet, and Yaser Yacoob. A framework for modeling appearance change in image sequence. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 660–667, 1998.

Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proceedings of the Seventh European Conference on Vision*, volume 4 of *LNCS 2350*, pages 707–720. Springer Verlag, 2002.

M. Bray, P. Kohli, and P. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *ECCV (2)*, pages 642–655, 2006.

Robert Collins, Yanxi Liu, and Marius Leordeanu. On-line selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631 – 1643, October 2005.

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

T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Image Understanding Workshop*, pages 1013–1026, 1996.

J. P. Costeira and T. Kanade. A multibody factorization method for independently moving-objects. *International Journal of Computer Vision*, 29(3):159–179, September 1998.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

P.F. Culverhouse and H. Wang. Robust motion segmentation by spectral clustering. In *British Machine Vision Conference*, pages 639–648, 2003.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

B.J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, February 2007.

A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, 2nd edition, 2003.

B. Georgescu, D. Comaniciu, T. X. Han, and X. S. Zhou. Multi-model component-based tracking using robust information fusion. In *2nd Workshop on Statistical Methods in Video Processing*, May 2004.

Z. Ghahramani and G.E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1996a.

Z. Ghahramani and G.E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto, 1996b.

Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

Amit Gruber and Yair Weiss. Factorization with uncertainty and missing data: Exploiting temporal coherence. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2003. ISBN 0-262-20152-6.

Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the EM algorithm. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 707–714, 2004.

Greg Hager and Peter Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 403–410, 1996.

Peter Hall, David Marshall, and Ralph Martin. Incremental eigenanalysis for classification. In *Proceedings of British Machine Vision Conference*, pages 286–295, 1998.

Peter Hall, David Marshall, and Ralph Martin. Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition. *Image and Vision Computing*, 20(13-14):1009–1016, 2002.

R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.

Michael Harville. A framework for high-level feedback to adaptive, per-pixel mixture of Gaussian background models. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proceedings of the Seventh European Conference on Vision*, volume 4 of *LNCS 2352*, pages 531–542. Springer Verlag, 2002.

L. Herda, P. Fua, R. Plankers, R. Boulic, and D. Thalmann. Using skeleton-based tracking to increase the reliability of optical motion capture. *Human Movement Science Journal*, 20(3):313–341, 2001.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Computational Neuroscience Unit, 2000.

Geoffrey E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–9, 1999.

Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In B. Buxton and R. Cipolla, editors, *Proceedings of the Fourth European Conference on Computer Vision (ECCV)*, volume 2 of *LNCS 1064*, pages 343–356. Springer Verlag, 1996.

Michael Isard and Andrew Blake. A mixed-state Condensation tracker with automatic model-switching. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998.

Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 415–422, 2001.

Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, October 2003.

G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2002.

Michael I. Jordan. *An Introduction to Probabilistic Graphical Models*. 200X.

Zia Khan, Tucker Balch, and Frank Dellaert. A Rao-Blackwellized particle filter for eigentracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

Adam G. Kirk, James F. O'Brien, and David A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2005. ISBN 0-7695-2372-2.

Marco La Cascia and Stan Sclaroff. Fast, reliable head tracking under varying illumination. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 604–608, 1999.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001.

Avraham Levy and Michael Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8): 1371–1374, 2000.

Jongwoo Lim, David Ross, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for visual tracking. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 793–800. MIT Press, 2005a. (Acceptance rate: 207/822, 25.18%).

Jongwoo Lim, David A. Ross, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for visual tracking. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *NIPS 17*. MIT Press, Cambridge, MA, 2005b.

Ruei-Sung Lin, David Ross, Jongwoo Lim, and Ming-Hsuan Yang. Adaptive discriminative generative model and its applications. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 801–808. MIT Press, 2005. (Acceptance rate: 207/822, 25.18%).

B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of International Joint Conference on Intelligence*, pages 674–679, 1981.

Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004.

Edward Meeds, David A. Ross, Richard S. Zemel, and Sam Roweis. Learning stick-figure models using nonparametric bayesian priors over trees. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

R. Neal and G. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

Ben North and Andrew Blake. Learning dynamical models using expectation-maximization. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 384–389, 1998.

Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*, February 16 2008. http://matrixcookbook.com.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *NIPS 17*, pages 1097–1104. MIT Press, Cambridge, MA, 2005.

Christopher Rasmussen and Gregory Hager. Joint probabilistic techniques for tracking multi-part objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16–21, 1998.

H.E. Rauch, F. Tung, and C.T. Striebel. Maximum likelihood estimates of linear dynamical systems. *AIAA Journal*, 3(8):1445–1450, 1965.

David A. Ross and Richard S. Zemel. Learning parts-based representations of data. *Journal of Machine Learning Research*, 7:2369–2397, Nov 2006.

David A. Ross, Jongwoo Lim, and Ming-Hsuan Yang. Adaptive probabilistic visual tracking with incremental subspace update. In T. Pajdla and J. Matas, editors, *Proc. Eighth European Conference on Computer Vision (ECCV 2004)*, volume 2, pages 470–482. Springer, 2004.

David A. Ross, Simon Osindero, and Richard S. Zemel. Combining discriminative features to infer complex trajectories. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

David A. Ross, Daniel Tarlow, and Richard S. Zemel. Learning articulated skeletons from motion. In *Workshop on Dynamical Vision at ICCV*, 2007.

David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77 (1–3), May 2008a. Special Issue on Machine Learning for Vision.

David A. Ross, Daniel Tarlow, and Richard S. Zemel. Unsupervised learning of skeletons from motion. In *Submitted for review*, 2008b.

S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2005.

Sam Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 626–632. MIT Press, 1997.

Sam T. Roweis. Gaussian identities. http://www.cs.toronto.edu/~roweis/notes/gaussid.pdf, July 1999a.

Sam T. Roweis. Matrix identities. http://www.cs.toronto.edu/~roweis/notes/matrixid.pdf, June 1999b.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

M. C. Silaghi, R.Plankers, R.Boulic, P.Fua, and D.Thalmann. Local and global skeleton fitting techniques for optical motion capture , modeling and motion capture techniques for virtual environments. In *Lecture Notes in Artificial Intelligence*, volume 1537, pages 26–40. Springer, 1998.

C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discrimintative density propagation for 3d human motion estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):814–827, July 2003.

Yang Song, Luis Goncalves, and Pietro Perona. Learning probabilistic structure for human motion detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 771–777. IEEE Computer Society, 2001. ISBN 0-7695-1272-0.

Erik B. Sudderth, Michael I. Mandel, William T. Freeman, and Alan S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *NIPS 17*, pages 1369–1376. MIT Press, Cambridge, MA, 2005.

Leonid Taycher, John W. Fisher III, and Trevor Darrell. Recovering articulated model topology from observed rigid motion. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1311–1318. MIT Press, 2002.

Leonid Taycher, Gregory Shakhnarovich, David Demirdjian, and Trevor Darrell. Condition random people: Tracking humans wth crfs and grid filters. Technical Report 2005-079, MIT-CSAIL, December 2005.

Y.W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, Dec 2003.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.

Kentaro Toyama and Andrew Blake. Probabilistic tracking in metric space. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 50–57, 2001.

Jaco Vermaak, Neil Lawrence, and Patrick Perez. Variational inference for visual tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 773–780, 2003.

Thomas Viklands. *Algorithms for the Weighted Orthogonal Procrustes Problem and Other Least Squares Problems*. PhD thesis, Ume University, Ume, Sweden, 2006.

Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1999.

Eric W. Weisstein. Digamma function. From MathWorld–A Wolfram Web Resource., a. http://mathworld.wolfram.com/DigammaFunction.html.

Eric W. Weisstein. Gamma function. From MathWorld–A Wolfram Web Resource., b. http://mathworld.wolfram.com/GammaFunction.html.

M. Welling. The Kalman Filter. http://www.ics.uci.edu/~welling/classnotes/papers_class/KF.ps.gz.

Max Welling, Michal Rosen-Zvi, and Geoffrey Hinton. Exponential family harmoniums with an application to information retrieval. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *NIPS 17*, pages 1481–1488. MIT Press, Cambridge, MA, 2005.

Oliver Williams, Andrew Blake, and Roberto Cipolla. A sparse probabilistic learning algorithms for real-time tracking. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 353–360, 2003.

Jingyu Yan and Marc Pollefeys. Factorization-based approach to articulated motion recovery. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005a.

Jingyu Yan and Marc Pollefeys. Articulated motion segmentation using ransac with priors. In *(ICCV) Workshop on Dynamical Vision (ICCV)*, 2005b.

Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part III*, 2006a.

Jingyu Yan and Marc Pollefeys. Automatic kinematic chain building from feature trajectories of articulated objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006b.