Towards Automatic Discovery of Human Movemes

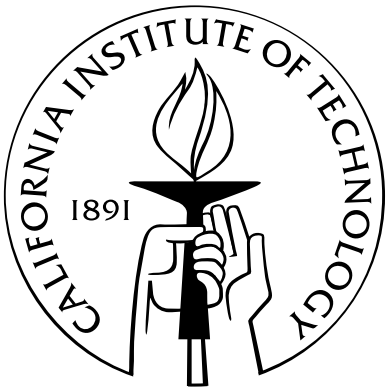
Thesis by

Claudio Fanti

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2008

(Defended January 30, 2008)

To my parents.

Acknowledgements

It is with great pleasure that I thank Prof. Pietro Perona for his help, support, and guidance throughout my research and studies. He has gone far beyond his role of advisor during these years. I feel very fortunate for all I have learned from him, both on a personal and a professional level.

I would like to thank Dr. Marzia Polito and Prof. Lihi Zelnik-Manor, with whom I had the pleasure to work on most of what I present in this thesis. They have been very understanding and have managed to get the best out of our efforts.

Thanks to Prof. Max Welling, Prof. Christof Koch, and Prof. Yaser Abu-Mostafa for their encouraging words and for agreeing to be part of my defense committee. I am particularly grateful to Max for the fruitful discussions and many useful suggestions: credit goes to him for first proposing the idea of using Loopy Belief Propagation for the labeling problem, and for some of the work and most of the ideas mentioned in Section 5.5.

A special thanks goes to my fellow members of the Computational Vision Lab and those of the Learning Group. They have made life in the lab a lot better during countless nights and many deadlines. Among them, I want to mention Marco for being a great roommate and friend, and for keeping me company at almost every meal I had in the last five years.

I really want to thank Ting for being next to me during the ups and downs of life. She has been such a relief in stressful moments by providing lot of affection, loads of patience,

and way more food than I should have eaten. I am looking forward to a life with her.

Finally, I want to thank my family for their endless love and support: without mom's discipline, dad's foresightedness, and a brother to live up to, I would have gotten nowhere near where I am.

Abstract

Consider a number of moving points, each attached to a joint of the human body and projected onto an image. Johannson showed that humans can effortlessly detect and recognize the presence of other humans from such displays. This is true even when some of the body parts are missing (e.g., because of occlusion) and unrelated clutter points are added to the display. Furthermore, subtle aspects like age range and gender, as well as the ongoing activity, can be inferred with a surprising degree of accuracy from such a seemingly scarce amount of information. We are interested in replicating some of these abilities in a machine.

We start by introducing a labeling and detection scheme in a Johannson-like display. Our method is based on a probabilistic representation of the positions and motion of body parts, which we use to calculate a likely interpretation of the scene by means of belief propagation techniques. We show how learning and inference can be done efficiently, and we provide an experimental validation of the method.

In the second part of our work, we present our position on the analysis of human behaviors. We hypothesize a hierarchical description of motion, which provides a natural interpretation of actions and activities as stochastic sequences of "atomic motions" or *movemes*. We take an initial step in that direction by illustrating how to learn a dictionary of movemes from the trajectories of body parts, which can be used to concisely represent the video for further analysis.

Contents

List of Figures

Chapter 1

Introduction

Human motion analysis is a very important and difficult problem in computer vision. When observing social interactions that take place in the surrounding environment, humans are, in general, the most important component.

The interest is further justified by the number of applications for which understanding people's actions and intentions is a central step. Among them, for instance, is monitoring people in airports or museums for security reasons. Detection of pedestrians is another example of application which is very attractive to the automotive industry for safety and autonomous navigation systems. Even the daily interaction with computers and appliances could be greatly improved by a more user-friendly interface (in a sense, a more passive one, where it is the machine that autonomously infers what we expect it to do).

Motion provides a large amount of information about humans and is very useful for social interactions. The goal of a human motion analysis system is to extract information about human motion from video sequences, in an attempt to produce a concise description of what is happening. Naturally, the first question to be addressed is whether or not there are humans in the scene. In case of positive answer, we are interested in knowing both their location and what action they are performing.

Our visual system perceives and analyzes human motion very rapidly; replicating this

Figure 1.1: People and Action. We show an image of people running and have superimposed some features with an arrow depicting their velocity.

ability in machines is one of the most challenging and ambitious goals of machine vision. Johannson's experiments [Joh73] show that humans find the instantaneous information on the position and velocity of a few features (such as the joints of the body, for instance) a sufficient cue to detect human presence and understand the gist of their activity. This still holds true even when clutter features are present in the scene, or some body-part features are occluded.

In this thesis, we present an approach to the task of detecting the presence of a human being, and labeling its body parts. We also develop a concise representation of its motion according to a dictionary of "atomic motions" (or *movemes*), which we learn from data without supervision. This is a basic step in trying to solve the broader problem of automatically recognizing people's actions and behaviors.

1.1 Features and Data

An important task that we will briefly discuss in Chapter 2 is that of identifying features in video sequences, describing their shape and appearance, and computing their velocity across frames by tracking their position in time. Although some instances of the problem are fairly well understood, and reasonably good solutions are available from the literature (see [TK91], for instance), a large number of factors influence the success of this component of the system, which we call the "front-end".

The resolution, or number of pixels, of the human body is probably the most influential aspect in determining the type of feature to be used; while a body height of 100 or more pixels is sufficient for a human to single out even portions of individual limbs, such as a fore-arm or an upper-leg, a 30-pixel-high man will not be suitable for a part-based decomposition, hence requiring a template-matching type of approache for its detection. Similarly, the resolution in time or frame-rate, as well as the amount of blur in the image due to the velocity of motion, pose a constraint on what features can be computed and which models are viable. Additional aspects to be considered, just to name a few, include the complexity of the video (due to textured background, for example), the variability in clothing colors and styles, or changes in viewpoint and scale, which are caused by the motion of the camera or the subject itself.

Although we have experimented with a few different approaches to feature detections, in most of our work we will assume that a number of features that are associated to the body have been detected and a description of their appearance has been computed. As no detector is perfect in practice, we will allow some of such features to be missing and admit that some are not at all associated with the body, but rather, originated from the

background.

1.2 Brief Survey of the Literature

The problems of detecting people and analyzing their behavior have received a great deal of attention in the last two decades. This effort is well motivated by the incredible variety of applications that a good solution would enable. Given the fair amount of material published in the field, providing a thorough survey is quite an endeavor of its own. In an attempt to set the context for our work, which we are going to present next, we settle for providing a few pointers to relevant approaches in the literature. In doing so we rely on the fantastic work of Forsyth and colleagues [FAI+05, IF07].

1.2.1 Detection of Humans in Videos and Static Images

The problem of detection answers the question of *whether there is* a human being in the scene or not. The output of a detection algorithm might also include a location (in various form, such as a (x, y) pair or a probability) where the person is believed to be, and possibly some sort of scale information, such as a bounding box around the detection. Although there are a number of axes along which one could categorize the different approaches, we will try to separate them into *template-based* and *part-based* methods.

1.2.1.1 Template-Based Methods

In the template-based methods an image is processed and several candidate locations are identified. In some instances, the locations could be all pixels in the image. A window around each location is (possibly encoded in some way and) further analyzed, as a whole, and a determination is made about whether it contains a person or not. Among others,

are the SVM-based work of Dalal and Triggs [DT05] and Papageorgiou and Poggio [PP00], the neural network approach of Zhao and Thorpe [ZT00], and the contour-based method of Gavrilla [Gav00]. A number of attempts have been made to represent not only 2D information but rather a set of frames simultaneously, giving rise to a space-time volume representation of motion. This approach is first found in Baker [Bak89], followed by Niyogi and Adelson [NA94], and applied to periodic motion in Cutler and Davis [CD00]. More recently, Viola et al. [VJS05] used motion features computed explicitly (and extremely efficiently), together with a cascade of boosted classifiers from their earlier face-detection scheme [VJ01].

1.2.1.2 Parts-Based Methods

The idea behind parts-based methods is to decompose the body into parts, solve the simpler task of localizing the parts, and finally work with the layout of the parts to establish human presence. The earliest application of this idea to vision problems is found in the work of Fischler and Elschlager [FE73]. More recently we have the discriminative methods of Ioffe and Forsyth [IF01] and of Ramanan et al. [RFZ05], where appearance of the parts is strongly relied on for detection, once a classifier for the part has been learned on the fly during an initialization pose. In the work of Felzenswalb and Huttenlocher [FH00] the emphasis is put on how to efficiently match the collection of parts to the model representing their layout. Song et al. [SGP01a, SGP01b] show how to learn the relationship among a number of point-features which are obtained by sparse optical flow [TK91]. Mikolajczyk et al. [MSZ04] build a parts detector for detection of faces, upper bodies, and legs. Liebe et al. [LSS05] extend the concept of part to patches and build a pedestrian detector. Mori et al. [MM02] [MREM04] represent the body as its set of joints and match the recovered

configuration against a set of stored poses. Agarwal and Triggs [AT04] use shape descriptors computed from the silhouette of the body.

1.2.2 Human Motion Analysis

Even assuming the existence of a method that can reliably detect people in each frame of a video sequence, we still need to figure out what humans are doing. The community has been extremely active in recent years. Besides a few methods that use motion in the form of spatio-temporal patterns—such as [EBMM03, BW97, BGS+05], and a few others we mentioned in the previous section—the majority of approaches use some form of explicit dynamical modeling, with the most popular choice being some flavor of hidden Markov model (HMM). Earlier attempts have modeled simpler motions, like tennis strokes [YOI92], pushes [WB95], and handwriting gestures [YXC97]. Feng and Perona [FP02] learn a vocabulary of quantized image shapes or "movelets", which they use as states for the HMM representing an activity. Brand et al. [BOP97] classify "Tai Chi" moves, Mori et al. [MSSS04] build a hierarchical representation of everyday gestures. A few alternatives based on linear dynamical systems (LDS) have also appeared in the literature. Bregler [Bre97] uses HMM and LDS to model activities at different levels of abstraction. Rohr [Roh97] represents walking with LDS. Black et al. [BYJ97] rely on paramerization of optical flow for the recognition of human activities. Del Vecchio et al. apply multiple LDSs for discrimination of drawing and reaching movements [DMP02, DMP03a, DMP03b]. Bissacco et al. [BCMS01, BCS07] recognize activities such as walking or jogging directly in the space of LDS. Pavlovic and Rehg [PR00] contrast the use of switching LDS and HMM in modeling of simple sequences of human motion.

1.3 Thesis Overview

In the next chapter we briefly review the process of extracting features from video sequences. Chapter 3 presents the problem of detection and labeling in moving light displays. We first review the work of Song et al. on triangulated models. Next, we present a few modeling extensions, such as the idea of global variables, and the introduction of more general structures to represent the connectivity among parts. We then describe how to do learning and inference in these models. We conclude the first part by providing some comparison and experimental validation in Chapter 4.

In Chapter 5 we introduce our hierarchical view of human motion. We start by describing a simple-minded approach to the discovery of movemes in video sequences, and show its performance on aerobic videos. We then motivate and present our work on modeling the dynamics of motion, and show how to do inference and learning with switching linear dynamical systems. We further show how the encoding of the dynamics, in conjunction with a clustering procedure, can be used to produce a small dictionary of movemes. The chapter is concluded by some more experimental results and a discussion. Our final remarks and some ideas for future work are in Chapter 6.

Chapter 2

Features Extraction and Front-End

There are a number of steps in designing a system that analyzes images or video sequences, and automatically detects objects and understands people's behavior. One crucial aspect involves the encoding, or representation, of the visual input in a tractable form, while preserving the relevant information needed for the task.

A number of techniques have been developed over the years which rely on finding "interesting" locations in the image by means of a *detector*, followed by an encoding of a small neighborhood of the image around the point into a vector or *descriptor*. In some situations, when the shape of the feature of interest is known, a more advanced detection scheme can be employed. For example, the detection of body parts has often been accomplished by looking for rectangular or box-like structures in the image, since the cylindrical shape of limbs (and, to some extent, of the torso) produces a pair of nearly parallel lines on the image plane.

Determining what type of detector-descriptor combination works best is not an easy task, since performance is greatly influenced by a number of factors. Among them are the quality and resolution of the image, the type of texture present, and, most importantly, the specific application. Although the literature is quite dense with techniques and recipes on how to identify and represent portions of an image, we will not review them here. Instead,

we briefly introduce a couple of approaches which have served as a front-end for our motion analysis work, which we present later on.

2.1 Point-Feature Detectors

A large number of the traditional point-feature detectors follow the same general scheme. The first step is to compute a saliency map that measures the local contrast in the image. Local maxima of the saliency map (typically corners or textured patches) are retained, since their location is more reliably determined in another image (or in the next frame). This is often the case, even if the image is taken from a slightly different viewpoint, or the human in the frame has moved to a nearby location. In order to provide some invariance to noise, only local maxima that exceed a given threshold are normally selected. If capturing structure at different scales is of interest, the process can be reapplied on an image which is obtained from the original one by smoothing and resampling.

An example of the saliency measure used in a few of the most popular detectors is the following

$$\mu = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.1}$$

where $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$ are the (spatial) partial derivatives of the image in the x and y direction.

The well known Harris detector [HS88] computes the local maxima of the map $\det(\mu) - 0.04 \cdot \mathrm{tr}^2(\mu)$. The Forstner detector [For86] selects features that are maxima of $\det(\mu)/\mathrm{tr}(\mu)$. The Lucas-Tomasi-Kanade (LTK) detector [TK91] averages μ over a small window around each pixel, and selects as features the points that maximize the smallest eigenvalue of the

resulting matrix. The common idea behind these three detectors is to select points where the image intensity has a high variability in both the x and the y directions.

The LTK detector is particularly useful when dealing with video sequences. In fact, a simple extension allows one to track features by matching the feature's local neighborhood in one image to the area surrounding the closest feature in the next image. This has the advantage that a velocity can be assigned to each feature point by computing the ratio of the spatial separation of the same feature in two consecutive frames, over the sampling period of the video sequence. The detection and labeling work of Song et al. [SGP00], as well as our own, is based on this type of feature.

2.2 Boxes Detectors

As we mentioned before, if the feature detection is aimed at identifying candidate body parts, better performance can be achieved by using an ad-hoc detector, which is geared toward the specific type of structure we are trying to find.

As part of our moveme discovery system we have developed a simple procedure to pre-process each frame of a video sequence, and identify candidate location with an approximately rectangular shape. This greatly reduces the number of features detected in each frame, and allows the matching of the models to be computed more efficiently.

Figure 2.1 illustrates the various phases involved in detecting boxes on a frame. We start on the left-hand side by feeding an image to a contour detector known as P_b or *probability of boundary*[1].

Boundaries represent the contour that separates two objects. Such contours have tradi-

[1]The P_b detector was originally developed by Martin et al. in [MFM04], to which we refer for implementation details and a thorough investigation of its performance.

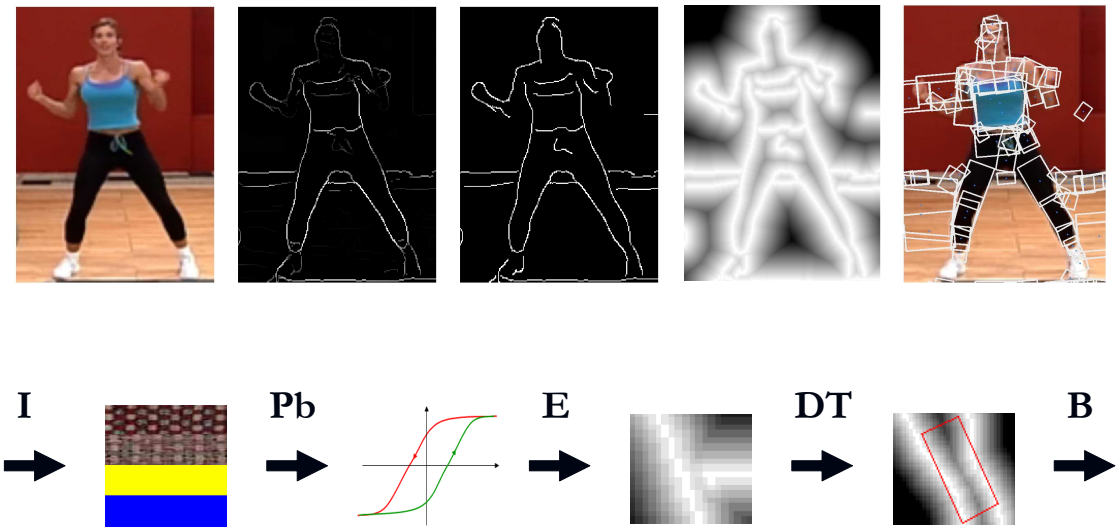


Figure 2.1: Box Detector. Starting with the original image, we compute a probability of boundary. We then perform a hysteretic thresholding which yields a binary map, to which we then apply the distance transform. Finally, the fitness to the distance transform image of each box's location and shape is evaluated, and the locally best-scoring boxes are retained.

tionally been discovered in a bottom-up fashion, by means of an edge detector which marks the abrupt change of some low-level property, such as brightness, from one pixel to the next. Alternatively, objects' boundaries can be identified by first recognizing all the objects that are present, thus computing a high-level interpretation of the scene. This is known as top-down processing.

The work of Martin et al. shows how multiple cues, such as brightness, color, and texture, can be combined into a single detector which computes the probability of each pixel being part of the boundary. The P_b detector is trained on a large set of natural images for which boundaries have been manually identified by several human subjects.

After we apply the P_b operator, the resulting image is the second from the left in Figure 2.1. The intensity represents the probability of the pixel belonging to a boundary. The next step consists of a hysteretic thresholding which yields a binary map. We compute the distance transform of the binary contour map by assigning to each pixel a brightness

level that is proportional to the distance from the closest boundary. The resulting image is the fourth of Figure 2.1. Finally, we compute a scoring function $\mathrm{s}(x, y, w, h, \alpha)$ which, for a given box centered at (x, y), with height h, width w and orientation α, assigns a score that measures the "boxiness" (i.e., the support) of the underlying image area, to a box of that shape, at that location. In determining the score of a box, the brightness of the pixel along the two "vertical" edges of the box, are considered. To actually obtain a set of box candidates, the function $s(\cdots)$ is locally maximized over small circular neighborhoods (a.k.a center-surround suppression), and the best-scoring box within is retained. To avoid detecting too many boxes, only a small set of (w, h, α)-triplets is considered. The side-effect of this choice is that the detector ceases to be scale and orientation invariant.

In our work on moveme discovery we use the procedure described to compute candidate locations for body parts.

2.3 Discriminatively Learned Features

One major issue which afflicts the features-detection schemes presented so far, and many others, is the rather poor discriminative power of the features. Unfortunately, computational constraints are such that most probabilistic models and algorithms of the type we will discuss in later chapters, can only handle a limited number of detections, and the performance is greatly affected by lots of clutter. Schemes like the Harris detector tend to generate features which are distributed all over the image, regardless of where the object of interest is.

A second source of nuisance in the fully bottom-up approaches to feature extraction has to do with the stability and good localization of the features. The criteria for their detection is rather simple, and relies entirely on low-level image information, such as the gradient. This implies that the features are most often originated by the combination of foreground

and background (that is, they are not due to the object of interest alone), and may not be found at all if the object is seen against a different background. Similarly, the location of the features themselves is most likely going to be determined by corners and lines, which in the case of highly deformable objects, such as the human body, tend to move around with respect to the object of interest, or even disappear.

For these reasons, it is important to have more part-specific detectors, which are trained to identify locations that are likely to belong to the object's part of interest. The training of a feature detector might require a labeled dataset, where the part of interest has been selected by an expert. As an alternative, a slightly less-specific level of supervision (such as a bounding box) combined with the hypothesis of consistency in appearance over the dataset, is sometime exploited in learning good features. It is generally a good idea to construct a dataset that is as diverse as possible, showing the different variants of the object's appearance (e.g., for humans it is desirable to train with different clothing, as well as different backgrounds). This enhances the detector's capability to generalize to new, unseen datasets.

The progress in the learning community, and the recent increase in computational power, have made the task more tractable by providing the machinery for automatically discovering which low-level image information is useful (or not) to the detection process. Example of successful systems are numerous in the literature. Among the earliest applications in vision, we mention the work of Viola and Jones [VJ01] for its relevance in the area of face detection, and that of Weber et al. [WWP00] for the innovative constellation model for object detection; among the most recent results is the empirical study on the performance of trained classifiers by Dollar et al. [DTTB07], which also includes a good review of various discriminative approaches to features detection.

2.4 Top-Down Effects on Features Detection

The features detection schemes we have talked about so far, follow a primarily bottom-up approach, that is, the low-level image information drives the detection and localization of the features. Since the exhaustive search over the entire image produces a large number of features, this poses a couple of problems. In fact, the substantial amount of features forces the inference machinery to deal with a much larger hypothesis space, therefore greatly increasing the running time. Additionally, the increased level of clutter makes it more likely to mistakenly match a part to the wrong detection.

On the other hand, if a high-level interpretation of the scene was available, either in the form of a guess for the overall location of the object, or as a probability distribution of where the parts of interest are located, then one could focus the *search* for features in a much more confined area of the image.

This is particularly true if we are looking for people in video sequences, since physical constraints exist on the motion a body part can undergo from one frame to the next. A model which incorporates dynamical information, can be used to predict what is the most likely location of the parts in the next frame, given the current state and position. This is especially helpful in the presence of prolonged occlusions, which would make the bottom-up task particularly difficult. One caveat of this idea is that even a small imprecision in the estimate of the current state would produce an erroneous prediction, which in turn causes the next estimate to fall even further from the unknown "true" value. This phenomenon is called *drifting* since the estimate tends to drift away from the underlying quantity it is trying to predict.

A possible way to alleviate the problem of drifting is to take the certainty over the

estimate of the state into account. When the model has a high confidence that the current estimate is correct and can be trusted, the feature detection relies on the prediction derived from that estimate and performs the search for features locally. If, instead, the estimate has a great deal of uncertainty, the feature search is performed in a nearly bottom-up fashion by exploring a larger area of the image. This combination of top-down and bottom-up can be made more precise by considering the prediction as a prior to the true location of the part, while the image measurements can be interpreted as a likelihood term. Standard inference algorithms can then be applied that estimate the most likely location of the feature. Although a few attempts have been made to implement a similar idea in tracking (see, e.g., [BI96]), we have seen relatively few attempts to combine this type of top-down/bottom-up approach to feature detection, with higher-level action understanding and interpretation. This is probably due to the already complex task of doing inference in the two components separately. Nevertheless, it is our belief that this is an interesting and important open question for which, even just a reasonable solution, would be very desirable.

Chapter 3

Detecting and Labeling People

3.1 Introduction

In this chapter we propose a solution to two very important problems in human-machine interaction: detection of a person and localization of its body parts. Our visual system seems to be able to solve these problems effortlessly, even from what is considered a very sparse and weak signal, as is the case for a Johannson's display [Joh73]. In Johansson's human perception experiments, the input to the human visual system is a set of dots; as soon as the dots start moving, we can get a vivid perception of the human, and immediately map each body part (such as hand, elbow, shoulder, knee and foot) to the correct dot. Furthermore, subtle aspects like age range and gender, as well as the ongoing activity, can be inferred with a surprising degree of accuracy even from such a seemingly scarce amount of information [MM94, CK77, DTLM96]. During this process, our visual system solves a hard combinatorial problem—the labeling problem: which dot should be assigned to which body part?

Our approach takes as input the instantaneous position and velocity of a few point-features, which are in correspondence with parts of the body, as well as a large number of distracting points arising from the background. We learn a probabilistic model from the

data, which describes the motion of the parts and their mutual relationships. The classifier, obtained by thresholding the likelihood of the data, makes the determination as to whether the optimal alignment of body parts to features supports the presence of a human at that location.

3.1.1 Graphical Models

Graphical models are a natural way of describing interaction among random quantities and an excellent conceptual representation to guide the implementation of inference schemas.

In their most general form, graphical models can be thought of as a machine that can answer queries regarding the values of a set of random variables, given the evidence that is known about some other set of variables. The beauty stands in the fact that this machinery is built combining information locally, and propagating it in agreement with the theory of probabilities in order to reach global consistency.

Due to the extensive literature available, we refer to [Lau96, Jor99, AM00, Mur02, YFW00, WF01, YFW05] for a thorough review of the subject of graphical models, and a number of algorithms based on the idea of propagating information and beliefs in those models.

3.1.2 Notation

In what follows we will use bold-face letters ($\mathbf{x}$) for random vectors and italic letters (x) for their sample values. The probability density (or mass) function for a variable $\mathbf{x}$ is denoted by $f_{\mathbf{x}}(x)$, while its expectation is written as $E_{f_{\mathbf{x}}}[x]$. An ordered set of indices $\mathcal{I} = [i_1 \dots i_K]$ (sometime written more concisely as a range $\mathcal{J} = [j_1 : j_2] = [j_1, j_1 + 1 \dots j_2]$) used as a vector's subscript has the intuitive meaning of $\mathbf{y}_{\mathcal{I}} = [\mathbf{y}_{i_1} \dots \mathbf{y}_{i_K}]$. When enclosed in squared

brackets $[\mathcal{I}]_s$ with a subscript s, and applied to a dimension of a matrix $V = [v_{ij}]$, it selects the s-dimensional members of the matrix along that dimension (e.g., $V_{[1:2]_4[1:2]_4}$ is the 8×8 matrix obtained by selecting the first two "stacks" of 4 rows and 4 columns).

3.2 The Labeling Problem

In the following sections we introduce the labeling problem in more formal terms, and discuss our modeling strategy and algorithm for its solution.

3.2.1 Problem Definition

We identify M relevant parts on the body, which intuitively correspond to the main joints. Given a display, each marked point (referred to as a *detection* or *observation*) is denoted by $y_i \in \mathbb{R}^4$ and is endowed with four values, i.e., $y_i = [y_{i,a}, y_{i,b}, y_{i,v_a}, y_{i,v_b}]^T$, corresponding to its horizontal and vertical positions and velocities. Given N observations, we would like to find the most probable assignment of a subset of the N detections to the M body parts.

For each display, we call $y = [y_1^T \dots y_N^T]^T$ the $4N \times 1$ vector of all observations on a frame, and we model each single observation as a 4×1 random vector $\mathbf{y}_i$. In general $N \geq M$; however, some or all of the M parts might not be present in a given display, that is, there might be no detection mapped to a part. To account for missing parts, we use a binary random variable $\boldsymbol{\delta}_i$, $i \in \{1 \dots M\}$ which indicates whether the i^{th} part has been detected or not.

For each part i, a discrete random variable $\mathbf{s}_i$, taking values on $\{1 \dots N\}$, is used to specify the correspondence of the part to a particular detection whose index is s_i. Since this makes sense only if the body part is detected, we assume by convention that $s_i = 0$ if $\delta_i = 0$.

A pair $\mathbf{h} = [\mathbf{s}, \boldsymbol{\delta}]$ is called a labelling *hypothesis*[1]. Any particular labelling hypothesis determines a partition of the set of indices corresponding to detections into foreground and background: $[1 \dots N]^T = \mathcal{F} \cup \mathcal{B}$, where $\mathcal{F} = [s_i : \delta_i = 1, i = 1 \dots M]^T$ and $\mathcal{B} = [1 \dots N]^T \setminus \mathcal{F}$. We say that $m = |\mathcal{F}|$ parts have been detected and $M - m$ are missing. Based on the partition induced on $\mathbf{s}$ by $\boldsymbol{\delta}$, we can define two vectors $\mathbf{s}^f = \mathbf{s}_{\mathcal{F}}$ and $\mathbf{s}^b = \mathbf{s}_{\mathcal{B}}$, each identifying the detections that were assigned to the foreground, and those assigned to the background, respectively. Finally, the set of detections $\mathbf{y}$ remains partitioned into the vectors $\mathbf{y}_{\mathbf{s}^f}$ and $\mathbf{y}_{\mathbf{s}^b}$ of the foreground and background detections, respectively.

The foreground and background detections are assumed to be conditionally independent given $\mathbf{h}$, meaning that their joint distribution factorizes as follows:

$$f_{\mathbf{y}|\mathbf{s}\boldsymbol{\delta}}(y|s\delta) = f_{\mathbf{y}_{\mathbf{s}^f}|\mathbf{s}\boldsymbol{\delta}}(y_{s^f}|s\delta) \cdot f_{\mathbf{y}_{\mathbf{s}^b}|\mathbf{s}\boldsymbol{\delta}}(y_{s^b}|s\delta).$$

Our goal is to find a hypothesis $\hat{h} = [\hat{s}, \hat{\delta}]$ such that

$$[\hat{s}, \hat{\delta}] = \arg\max_{s\delta} \{f_{\mathbf{y}_\mathbf{s}|\mathbf{s}\boldsymbol{\delta}}(y_s|s, \delta)\}. \tag{3.1}$$

Now that we have expressed the problem in a general form, we go on in the following sections introducing different variants of it, and the techniques to solve it. We start by reviewing some prior work by Song et al. [SGP00] on triangulated graphs and their dynamic programming approach, and then continue with our contribution, both on the modeling and the algorithmic side.

[1] In the remainder of this thesis we will refer to either the pair $(\mathbf{s}, \boldsymbol{\delta})$ or the $\mathbf{h}$ interchangeably.

3.2.2 Conditional Independence and Computational Complexity

Let us start by focusing our attention on a subset of the variables in our problem and their density, namely the variables $\mathbf{x}_1 \dots \mathbf{x}_M$, which represent the M body parts. To further simplify the problem, let us assume for now that all parts have been observed in the display and we are to figure out which part matches which detection. An immediate solution is to evaluate the density of the parts on all possible assignment of the M parts to (a subset of) the detections, and retain the most promising assignment, according to the probability measure. Although correct in principle, the proposed brute-force approach is clearly infeasible for any problem of interest as it would require the evaluation of $\mathcal{O}(N^M)$ hypothesis. If, however, a subset of the parts were moving independently of the others, one could reduce the number of hypothesis to score to $\mathcal{O}(N^{M'})$, $M' < M$, by noticing that the most probable assignment could be determined independently for the two sets. In the degenerate case where each part is moving independently of every other part, we have $M' = 1$ and the complexity is linear in the number of detections.

The idea of using independence among parts was introduced by Song et al. in [SGP00, SGP01a, SGP01b], and is further explored by us in this work. We will now briefly review it in the context of graphical models.

Following the graphical models notation, we assign to each variable in the problem a node in a directed acyclic graph. Links between nodes indicate a relationship of dependence between the variables. This dependence can be visualized, locally, as a *child-parents* relationship; ignoring the rest of the graph, we can define a family as a node (the child) and all the nodes linking towards that child (its parents). Examples of families within a graph are shown in Figure 3.1.

For a given structure of the graph a number of conditional independences are imposed

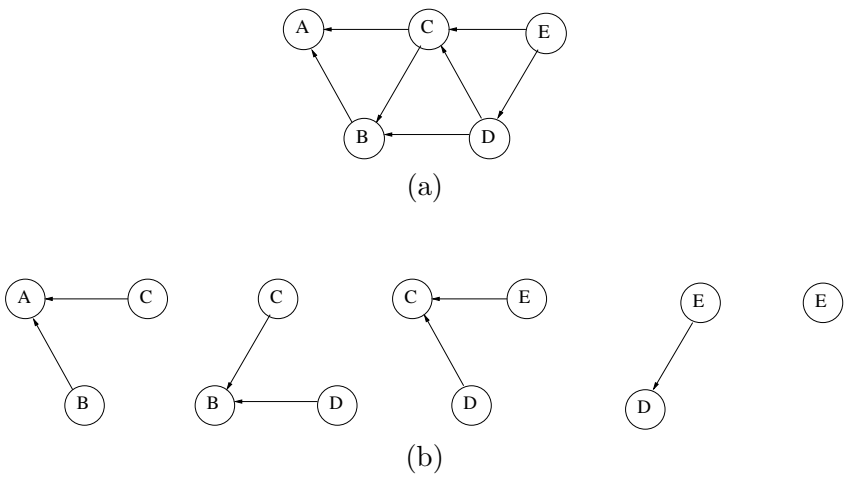


Figure 3.1: Conditional Independence. We show a graphical model representing a probability density over five variables (a) and the five families into which the density can be factorized (b).

on the probability density that the graph represents. These can be "read out" by inspection of the graph, allowing us to rewrite the density in its factorized form. This is done by going through each node in the graph and listing one factor per family.

In the example of Figure 3.1, the factorization into families becomes

$$f(ABCDE) = f(A|BC)f(B|CD)f(C|DE)f(D|E)f(E). \tag{3.2}$$

More generally, if i is the index of a node in the graph and π_i is the set of indices of x_is parents, we have the following factorization:

$$f(x_1 \dots x_n) = \prod_{i=1}^{n} f(x_i|x_{\pi_i}). \tag{3.3}$$

The benefits of having a factorized form are enormous. Let us suppose, for example, that we want to maximize the density $f(A, B, C)$ with respect to its three variables. Let us

further assume that the density factorizes as follows:

$$f(ABC) = f(A|B)f(B|C)f(C). \tag{3.4}$$

We can then write

$$\begin{aligned} \max_{ABC} f(ABC) &= \max_{ABC} [f(A|B)f(B|C)f(C)] \\ &= \max_{AB} \left[f(A|B) \max_{C} [f(B|C)f(C)] \right] \\ &= \max_{AB} [f(A|B)g(B)] \end{aligned} \tag{3.5}$$

and notice that, while the original maximization involved optimazing with respect to three variables, we only had to maximize functions of two variables and then correctly combine the results together (i.e., multiply them) to obtain the same answer. In [AM00] it is observed how this process can be applied not only when the two operations involved are *max* and *product*, but also with *sum* and *product* (used to efficiently compute marginals of a density), as well as with a number of other operations and sets which have the appropriate algebraic structure.

Going back to the problem of matching parts with observations, it is now clear how the degree of independence among the parts determines the computational cost of maximizing (or marginalizing) the density. It has been shown (see, e.g., [Pea88]) that the complexity of such operations is exponential in the size of the largest clique in the graph.

3.2.3 Triangulated Model

Although it is hardly ever the case that any two parts of the body are moving independently, computational reasons force us to approximate the description of the data with models that

are tractable. As we have seen, one such way is to assume some degree of independency among the parts, so that efficient inference is possible. In [SGP00] Song et al. have explored the use of conditional independence assumptions in the context of detecting and labeling humans. In their work, they represent conditional independence by means of *triangulated decomposable* undirected graphs, which they define as

1. either a clique of size three or
2. a collection of cliques of size three, for which there exist an elimination order of the vertices, such that when a vertex (and the two edges connecting it to its clique) is deleted,
   - it does not belong to any other clique, and
   - the remaining graph is still a triangulated decomposable graph.

Although multiple elimination orders might be possible, for a given graph, in Song's work, the order is fixed to a specific one.

Once the elimination order has been fixed, it is possible to build an "equivalent" directed acyclic graph from the undirected triangulated decomposable graph. This is done by visiting each vertex, according to the elimination order, and directing toward it the two edges of the *only* clique such vertex belongs to. The directed graph thus obtained represents the same set of conditional independences as the original undirected graph. The computational complexity associated with a triangulated model is $\mathcal{O}(N^3)$, since each factor in the decomposition is of size three. Alternative independence assumptions, such as tree-like graphs with pair-relationships, rather than triplets, have been explored by Song et al; however, the triangulated graphs are presented as the best trade off between accuracy of the representation and computational complexity.

3.2.4 More General Connectivity Models

Encouraged by the promising experiments of [SGP00], we have explored the trade off between computational cost and likelihood of the data when using graphical models with more complex structure.

The first aspect we investigate is the level of independence imposed among the body parts. As we have already mentioned, given a finite dataset, no two parts are in general independent of each other, given all the other parts. However, due to computational consideration, we are forced to introduce some conditional independences.

The problem of selecting which (in)dependence relationships to impose amounts to selecting a particular structure for the graphical model representing the probability density of body parts. It has been shown [Chi96] that automatically identifying the optimal structure (the one maximizing the likelihood of the data) is a NP-hard problem. Nevertheless, heuristics can be employed to find a locally optimal solution.

One way of quantifying the level of independence among parts is to limit the "fan-in" of each child by allowing at most K parents. This has the advantage of capping the size of the largest clique (i.e., family) to $K+1$, hence setting the computational complexity of inference in the graph to $\mathcal{O}(N^{K+1})$. In [SGP00] K is fixed to two and the computation is cubic in the number of observations. In our work we have observed that having higher values for K (thus representing more dependencies among the body parts, at the price of more computation) does not significantly increase the performance of the system.

We have mentioned in Section 3.2.3 how, in addition to having a fan-in of two, a triangulated decomposable graph must also admit the existence of an appropriate elimination order. It turns out that this definition is equivalent to the more widely known *junction tree* property. Namely, if a graph is triangulated decomposable, then the graph of its cliques is

a tree and the cliques have cardinality three. Furthermore, for any two cliques U and V in the clique-graph the (only) path connecting U and V is comprised of cliques, all of which contain the nodes belonging to $U \cap V$.

Although Song et al. define and use the terms "decomposable" and "triangulated" in a different way than most authors, we are ultimately interested in the fact that the type of graphs they use are exactly those that admit a junction tree with maximal cliques of size three. As a consequence, their setup allows efficient *exact* inference in cubic time. A natural question to ask is whether the tree shape of the clique graph is a necessary compromise. After all, there is a potential for a more accurate description of the data if we relax the requirement for the clique graph to be a tree and allow for a more general connectivity. Clearly this would come at a price, since only approximate inference could be performed, and techniques such as exact belief propagation, or dynamic programming, would not apply any longer. Nevertheless, encouraged by enough empirical evidence [MWJ99] justifying the application of Pearl's belief propagation to loopy graphs, we explored this extension, which we describe in Section 3.3.3.

3.3 Inference

In Section 3.2.1 we have introduced an instance of the labeling problem which was presented in [SGP00]. In order to provide a complete definition of the problem we will now specify in more detail the graphical model and probability densities used by Song et al., as well as their dynamic programming algorithm for the most likely labeling assignment.

3.3.1 Dynamic Programming

We recall that $\mathbf{x}$ represents the set of body parts we wish to model, while $\mathbf{s}$ and $\boldsymbol{\delta}$ are the labeling and detection variables, which match parts to a subset of the observations $\mathbf{y}$. Figure 3.2 (c) shows a plate graphical model depicting the interaction among the variables.

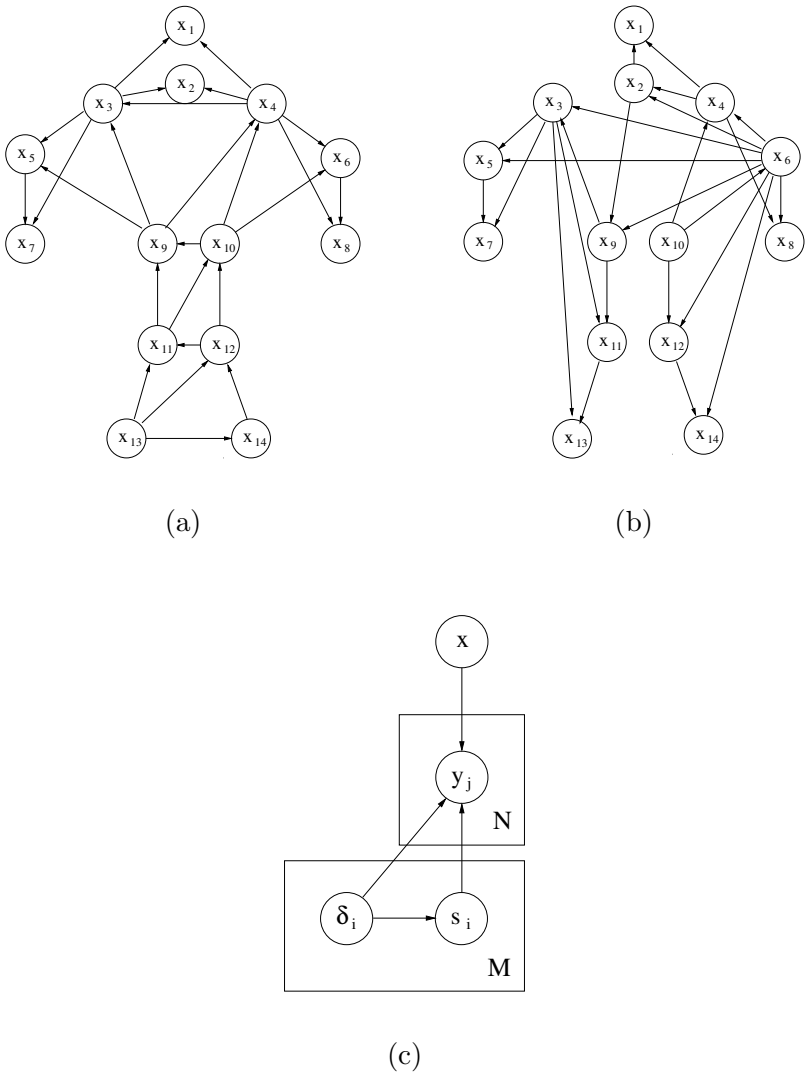


Figure 3.2: Body Decomposition. We show two examples of graphical models for the body parts variables $\mathbf{x}_1 \dots \mathbf{x}_M$. (a) shows a hand-crafted decomposition made by an expert, and (b) shows another decomposition automatically computed. Both are obtained from [SGP00] by fixing an elimination order of the nodes, and setting the directions of the arrows according to the procedure detailed at the end of Section 3.2.3. In (c) we show the full plate-model representing the interaction between the M body parts $\mathbf{x}_1 \dots \mathbf{x}_M$, the N observations $\mathbf{y}_1 \dots \mathbf{y}_N$, and the M labeling variables $(\mathbf{s}_1, \boldsymbol{\delta}_1) \dots (\mathbf{s}_M, \boldsymbol{\delta}_M)$.

The outgoing arrow that connects the node labeled $\mathbf{x}$ to the detections $\mathbf{y}_j$ indicates that each part $\mathbf{x}_i$, in principle, could have been "generated" by any one of the $\mathbf{y}_j$ observations. The labeling nodes $\mathbf{s}_i$ and $\boldsymbol{\delta}_i$, once known, determine which of these matches is the correct one.

The factorization of the joint probability density can be immediately derived from the graphical model:

$$f_{\mathbf{xys\delta}}(x, y, s, \delta) = f_{\mathbf{x}}(x) \prod_{j=1}^{N} f_{\mathbf{y}_j|\mathbf{xs\delta}}(y_j|x, s, \delta) \prod_{i=1}^{M} \left[f_{\mathbf{s}_i|\boldsymbol{\delta}_i}(s_i|\delta_i) f_{\boldsymbol{\delta}_i}(\delta_i) \right]. \tag{3.6}$$

We start by examining the second factor on the right. Either one of two possibilities can occur; if y_j has been mapped to a part x_i, i.e., $\delta_i = 1$ and $s_i = j$, then

$$f_{\mathbf{y}_j|\mathbf{xs\delta}}(y_j|x, s, \delta) = \mathbf{1}\{y_{s_i} = x_i\}; \tag{3.7}$$

otherwise, the observation y_j must have come from the background distribution, which is uniform over the volume V

$$f_{\mathbf{y}_j|\mathbf{xs\delta}}(y_j|x, s, \delta) = \frac{1}{V}. \tag{3.8}$$

By combining all of the observations we obtain their joint density, which simplifies as follows:

$$\begin{aligned} f_{\mathbf{y}|\mathbf{xs\delta}}(y|x, s, \delta) &= \prod_{j=1}^{N} f_{\mathbf{y}_j|\mathbf{xs\delta}}(y_j|x, s, \delta) \\ &= \prod_{i \in \mathcal{F}} \mathbf{1}\{y_{s_i} = x_i\} \prod_{k \in \mathcal{B}} \frac{1}{V} \\ &= \mathbf{1}\{y_{s^f} = x_{\mathcal{F}}\} \left(\frac{1}{V}\right)^{M-|\mathcal{F}|} \left(\frac{1}{V}\right)^{N-M}. \end{aligned}$$

If we then marginalize (3.6) over the hidden variables $\mathbf{x}_i$, we obtain

$$\begin{aligned} f_{\mathbf{ys\delta}}(y,s,\delta) &= \int f_{\mathbf{xys\delta}}(x,y,s,\delta)dx \\ &= \prod_{i=1}^{M}\left[f_{\mathbf{s}_i|\boldsymbol{\delta}_i}(s_i|\delta_i)f_{\boldsymbol{\delta}_i}(\delta_i)\right]\int f_{\mathbf{x}}(x)\prod_{j=1}^{N} f_{\mathbf{y}_j|\mathbf{xs\delta}}(y_j|x,s,\delta)dx \\ &= \prod_{i=1}^{M}\left[f_{\mathbf{s}_i|\boldsymbol{\delta}_i}(s_i|\delta_i)f_{\boldsymbol{\delta}_i}(\delta_i)\right]\left(\frac{1}{V}\right)^{N-|\mathcal{F}|}\int f_{\mathbf{x}_\mathcal{F}}(x_\mathcal{F})\mathbf{1}\{y_{sf}=x_\mathcal{F}\}dx_\mathcal{F} \\ &= \left(\frac{1}{V}\right)^{N-M}\prod_{i=1}^{M}\left[f_{\mathbf{s}_i|\boldsymbol{\delta}_i}(s_i|\delta_i)f_{\boldsymbol{\delta}_i}(\delta_i)\right] f_{\mathbf{x}_\mathcal{F}}(y_{sf})\left(\frac{1}{V}\right)^{M-|\mathcal{F}|} \end{aligned} \tag{3.9}$$

where $f_{\mathbf{x}_\mathcal{F}}$ is the marginalized version of $f_{\mathbf{x}}$ over the missing parts.

In [SGP00], it is assumed that $f_{\boldsymbol{\delta}_i}(\delta_i=1) = f_{\boldsymbol{\delta}_i}(\delta_i=0) = 0.5$. Additionally, $f_{\mathbf{s}_i|\boldsymbol{\delta}_i}(s_i|\delta_i) = \frac{1}{N}$, since before observing any detection, each of them has an equal chance of being generated by the i-th body part. If we fix the number N of detections in a given frame, then we can rewrite (3.9) as such:

$$f_{\mathbf{ys\delta}}(y,s,\delta) \quad \propto \quad f_{\mathbf{x}_\mathcal{F}}(y_{sf})\left(\frac{1}{V}\right)^{M-|\mathcal{F}|}. \tag{3.10}$$

An interesting interpretation of (3.10) is the following: for a given frame, we can produce any labeling hypothesis we wish; we can vary the number of detected parts, by acting on $\delta_1 \ldots \delta_M$, and we can change the mapping $s_1 \ldots s_M$ between parts and detections. Whenever a part i is declared missing, a uniform factor $\frac{1}{V}$ "compensates" for it in the density.

Since ours is a probabilistic model, we can compare any two hypothesis by computing their likelihood with respect to the distribution. However, we have already noted in Section 3.2.2 that the exhaustive search over all hypotheses is infeasible, and that we need a more efficient way to compute the optimal solution. As our first step in that direction, we briefly review an algorithm from [SGP00], which takes advantage of the factorized form of

$f_{\mathbf{x}}$ to compute the most likely labeling hypothesis in polynomial time.

Assuming a decomposition similar to that of Figure 3.2 (a) or (b), we can list the M families in the graph as follows

$$\begin{aligned} f_{\mathbf{x}}(y_s) &= \prod_{i=1}^{M} f_{\mathbf{x}_i|\mathbf{x}_{[\pi_i]}}(y_{s_i}|y_{s_{[\pi_i]}}) \\ &= \prod_{i=1}^{M-2} \Psi_i(L_i) \end{aligned}$$

where the sets $L_i = [s_i, s_{\pi_i}]$ identify the indices of detections matched to the parts in each clique, $\Psi_i(L_i) = f_{\mathbf{x}_i|\mathbf{x}_{[\pi_i]}}(y_{s_i}|y_{s_{[\pi_i]}})$, and $\Psi_{M-2}(L_{M-2})$ is a joint density (rather than a conditional density) over the last three parts. This is the case, since the last two families, which derive from the last triangle, have either one or no parents. We can multiply them into the $M-2$ family, thus obtaining the joint $f_{\mathbf{x}_{M-2}\mathbf{x}_{M-1}\mathbf{x}_M}(y_{s_{M-2}}, y_{s_{M-1}}, y_{s_M})$.

So far we have ignored the possibility that some of the parts may be undetected in a frame. This is taken into account in [SGP00] by setting the appropriate $\Psi_i(\cdot) = \frac{1}{V}$ (with the obvious special handling of $\Psi_{M-2}(\cdot)$, since it represents not one, but three parts).

Having decomposed the density in a product of smaller functions, we can apply the trick illustrated in (3.5). Let us rewrite $L_i = [A_i, B_i, C_i]$ where, with the exception of the last function in the elimination order, B_i and C_i are the indices of the conditioning variables.

We start by defining and maximizing the first cost function Q_1 with respect to A_i, i.e.,

$$\begin{aligned} Q_1(A_1, B_1, C_1) &\triangleq \Psi_1([A_1, B_1, C_1]) \\ T_1(B_1, C_1) &= \max_{A_1}[Q_1(A_1, B_1, C_1)] \\ I_1(B_1, C_1) &= \arg\max_{A_1}[Q_1(A_1, B_1, C_1)]. \end{aligned}$$

Next, we build the second function Q_2. If Ψ_2 and Ψ_1 are "neighbors", that is, if $|L_2 \cap [B_1, C_1]| = 2$, then we set

$$Q_2(A_2, B_2, C_2) \triangleq \Psi_2([A_2, B_2, C_2])T_1(X, Y),$$

where X and Y are equal to A_2, B_2, or C_2, depending on which two of these three parts coincide with B_1 and C_1. If instead Ψ_2 and Ψ_1 are not neighbors, $Q_2 \triangleq \Psi_2$.

We maximize Q_2 with respect to A_2 and store the result as follows:

$$\begin{aligned} T_2(B_2, C_2) &= \max_{A_2}[Q_2(A_2, B_2, C_2)] \\ I_2(B_2, C_2) &= \arg\max_{A_2}[Q_2(A_2, B_2, C_2)]. \end{aligned}$$

At step t, we multiply together the function Ψ_t and all the partial results T_j of its neighbors, up to $j < t$. Finally, when $t = M - 2$ we maximize $Q_{M-2}(A_{M-2}, B_{M-2}, C_{M-2})$ to obtain A^*_{M-2}, B^*_{M-2}, and C^*_{M-2}. These are guaranteed to be the maximizing choices for the three parts they represent. We then propagate backward, piecing together the solution: at step t we set $A^*_t = I_t(B^*_t, C^*_t)$, that is, we determine the maximizing choice for A_t by using the optimal choice for the two parts B_t and C_t which are in the same clique; this can be done thanks to the fact that those two parts are shared by other cliques (where they are As) and their value was set in the previous steps of the back-tracking.

The reassurance that the algorithm is correct comes from noticing how the definition of triangulated graph requires that a node belong to only one clique (at the time of its elimination). Thus when we maximize with respect to A_i and store the optimal choices in $T_i(B_i, C_i)$, we are guaranteed that A_i won't appear in any other function that needs to be maximized; therefore, any choice made regarding A_i up to that point, will remain optimal

for the remainder of the algorithm.

To achieve translation invariace, Song et al. model the relative positions of parts within a clique, rather than their absolute values. This does not change the essence of the algorithm, since the additional coupling between parts is local to a family and preserves the structure of the graph.

3.3.2 Belief Propagation

Now that we have reviewed the work in [SGP00], we go on to show how their dynamic programming algorithm is in fact a particular case of a more general class of inference algorithms known as *belief propagation*.

The problem of probabilistic inference is that of computing the expectation of a function $g(\mathbf{h}_L)$ defined over a subset L of the variables $\mathbf{h}_1 \dots \mathbf{h}_M$, with respect to the density $f_{\mathbf{h}_1 \dots \mathbf{h}_M}$, i.e.,

$$\mathbb{E}_{f_{\mathbf{h}_1 \dots \mathbf{h}_M}} \left[g(h_L) \right]. \tag{3.11}$$

This requires us to compute the marginal $f_{\mathbf{h}_L}(\cdot)$ of $f_{\mathbf{h}}(\cdot)$. Furthermore, if we had to compute the expectations for multiple functions, the process would quickly become computationally expensive, as we would have to marginalize the density over a different set of variables each time. Luckily, the belief propagation algorithm comes to our rescue. As we will see in a moment, its message-passing procedure simultaneously and efficiently computes marginals on each clique of variables in the factorization; the computational saving is made possible by reusing the intermediate result of one computation in the following ones[2].

[2]We should mention that although we have been focusing on marginalization, in general we can replace the "*sum*" operation involved in marginalizing with the "*max*". This would maximize the density for each clique, with respect to all but the variables in that clique. Other operators, beside max and sum, can also be used. See [AM00] for a more thorough introduction to the topic.

Let us assume we have a graphical model $\mathcal{G}$, which characterizes the probability density function over the variables $\mathbf{h}_1 \dots \mathbf{h}_M$. Let us also assume that the connectivity in $\mathcal{G}$ is sparse, so that the density factorizes, and can be written as

$$f_{\mathbf{h}_1 \dots \mathbf{h}_M}(h_1 \dots h_M) = \prod_{i=1}^{M} f_{\mathbf{h}_i | \mathbf{h}_{\pi_i}}(h_i | h_{\pi_i}), \tag{3.12}$$

where π_i is the parent set of i. This is the typical situation when dealing with a direct graph. In the most general form the density can be a normalized product of positive (and integrable) functions called *potentials*

$$f_{\mathbf{h}_1 \dots \mathbf{h}_M}(h_1 \dots h_M) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(h_C), \tag{3.13}$$

where C is the set of indices of the variables contained in a family or clique (the *domain* of the potential), and $\mathcal{C}$ is the set of all cliques.

We build a graph $\mathcal{J}$ with the cliques of $\mathcal{G}$ as vertices, and an edge between any two cliques B and C when $B \cap C \neq \emptyset$. $\mathcal{J}$ is called the *lique graph* of $\mathcal{G}$. Although, in general, the graph $\mathcal{J}$ has arbitrary structure, under certain conditions on the connectivity of $\mathcal{G}$, $\mathcal{J}$ has a tree structure and it is called the *clique tree*.

A clique tree $\mathcal{J}$ is said to have the *junction property* if, for any two cliques B and C, every clique A, on the unique path connecting B to C, is such that $(B \cap C) \subset A$. Such a clique tree $\mathcal{J}$ is called the *junction tree* of $\mathcal{G}$.

For each edge $B \rightarrow C$ of the junction tree $\mathcal{J}$, we define a *message* over the variables $h_{B \cap C}$

$$m_{BC}(h_{B \cap C}) = \sum_{h_{B \setminus C}} \Psi_B(h_B) \prod_{A \rightarrow B \in \mathcal{J},\ A \neq C} m_{AB}(h_{A \cap B}). \tag{3.14}$$

The definition is clearly recursive, the base case being the message departing from leaf clusters of the junction tree.

To compute the desired marginals, both a forward pass (from the leaves to the root of the tree) and a backward pass (towards the leaves) must be completed. Once all the messages have been computed, for each cluster we define its potential

$$b_C(h_C) = \Psi_C(h_C) \prod_{B \to C \in \mathcal{J}} m_{BC}(h_{B \cap C}) \tag{3.15}$$

which we call the *cluster belief*. Given the belief on a cluster, we can easily obtain the marginal density $f_{\mathbf{h}_C}(h_C)$ by normalization.

At this point, the similarity between Song's dynamic programming algorithm and the message passing procedure starts to emerge. By using the "max" operation, rather than marginalizing, and noticing how clique potentials and messages play the role of the Q and T quantities of Song, we can see how the two algorithms are *essentially* the same.

Although the complexity is of the same order of magnitude for both algorithms, a history of maximizing choices is being kept in I_t by the dynamic programming procedure, which, in practice, improves on the computational cost by avoiding the backward pass of messages.

3.3.3 Loopy Belief Propagation

The message-passing procedure described in the previous section achieves its efficiency and guarantee of convergency by imposing certain structural constraints on the underlying probability density. We have already mentioned that, depending on the specific application, such constraints might be well justified, or they might even be intrinsically part of the problem. Other times, instead, they are simply a modeling approximation dictated by the demand for

fast computation, such as in the case of the triangulated decomposable graph of [SGP00]. In this section we relax the requirements for the graphical model to be decomposable, thus expanding the class of admissible graphs.

We now briefly review the issues involved in doing (approximate) inference on these more general graphs. In Section 3.4.1 we describe a procedure for learning their structure from labeled data.

Let us start by recalling the recursive definition (3.14). A message from a node B to a neighbor node C is computed by combining B's potential with all messages coming into B from its neighbors (excluding C). The recursion starts at the leaves of the tree, which by definition, have no incoming edges. Once all the messages have been computed according to the schedule induced by this recursion, the tree is said to be *calibrated* and we can compute the beliefs on each node as in (3.15).

When applying the message passing procedure to a loopy graph, the first issue we encounter is with the definition of the messages themselves. Since the resulting junction graph might not have leaves at all, the recursion's base step doesn't apply and we are left with an ill-posed procedure. A solution to the problem is to initialize all the messages to 1 and proceed with the update rule in (3.14) as before. This has the advantage that if the graph was indeed a tree, after we exchange twice as many messages as there are edges in the tree, the messages are guaranteed to converge to those produced by the original recursive schedule. The procedure thus described is called *loopy belief propagation (LBP)*.

A second and more depressing issue arising from the use of loopy graph has to do with the convergence of the message-passing algorithm. Although we are able to propagate the messages, and we can compute the belief at each node, we have no theoretical reassurance that this procedure comes to a point of equilibrium. Messages might alternate over different

states and never settle into a final configuration. This is the case even when partial or *soft updates* are implemented, where the new value of the message is a combination of its prior value and new value proposed by the (3.14). Nevertheless, empirical evidence shows that numerous problems of practical interest can be solved with this method, most notably the so called "turbo codes".

Several fundamental results have been reported in recent years which make LBP an appealing procedure (see, e.g., [MWJ99, YFW00, YFW05]). Among them, the work in [WF01] on the max-prod version of the algorithm (when the *max* operation is used in place of *sum*) is possibly the most encouraging. There, Weiss et al. showed that, although LBP might not always converge, when it does, the solution found is often very close to the global optimum. Additionally, the solution obtained is a "neighborhood maximum", that is, changing the solution on nodes of an arbitrary set of disconnected trees and single loops worsens the solution.

3.3.4 Global Variables

In our discussion we have so far sidestepped the problem that the body can undergo horizontal and vertical translations from one frame to the next. Approaches that model the absolute position of the parts with respect to the frame are clearly infeasible: since we cannot predict where the human will appear within the field of view of the camera, we need to build into the system some form of invariance with respect to translation. In [SGP00] this issue is dealt with in a "local" manner, by replacing the absolute positions with the relative location of two parts with respect to the third. This is a simple mechanism which allows the triangular clique to float over the frame, while still characterizing the mutual relationship among the parts it represents. The main advantage of dealing with translations locally is

that the structure of the graphical model remains unchanged, the conditional independences are maintained and the same efficient inference algorithms can be applied.

Although effective, local invariance presents some drawbacks. In particular, when the reference part is missing, we are unable to compute relative positions and have to resort to some heuristic to be able to compute a probability for the remaining stray parts; this is a necessary step since we want to score this hypothesis against others.

More importantly, when several parts are not detected, there is a risk for the model to break into multiple disjoint components. In fact, given two sets of parts, nothing prevents the matching of the first set of parts to observations that are far apart in the frame from the observations matched to the second set of parts: there could be a situation in which, e.g., the lower and upper portions of the body end up capturing two separate regions of the image as their most likely set of detections.

The missing ingredient here is the notion that although deformable, the human body as a whole is a well-localized entity: even though, at the level of its parts, the choice of which observation matches which part is a somewhat local decision, the collection of all parts must "make sense" as a body, and needs to be located within a bounded region.

We think of the "location of the whole body" as a *global* property, that is, a property which belongs to (and affects) all the parts at once. This is in contrast to the geometry (or the appearance) of a small group of parts within a limb which has, in principle, little to do with that of parts in another limb, and is thus a *local* property.

Much like the location of the body, which we call its *centroid*, the scale is also a global aspect which should be modeled in a centralized manner. As we will see in Chapter 5, the idea of global variables will be a key aspect of our approach to modeling action and activities.

Since we would like to simultaneously capture "global" and "local" properties, our approach to describing the motion of a human being results in a hybrid model. This presents additional computational challenges; among them is the difficulty of doing inference in a graphical model where the global variables influence every node, making any form of independence among the parts vanish due to the global coupling.

We now reformulate the model, with the inclusion of the global variables, and follow with our approach to learning its parameters and doing inference.

For the sake of simplicity, we limit ourselves to two global variables:

- **centroid:** a real-valued quantity $\boldsymbol{\theta} \in \mathcal{R}^2$ indicating the location of the body with respect to the frame's reference system, and

- **phase:** a discrete value $\boldsymbol{\phi} \in [1, \ldots n_{\boldsymbol{\phi}}]$ acting as an index over a set of possible poses or "*phases*" of the motion.

Since we are focusing on translation invariance, we focus on doing inference in the presence of the centroid; however, we present an extended version of our hybrid model that includes the phase, in anticipation of our work on movemes which we will present later.

Let us start by introducing the new graphical model in Figure 3.3, which we have enhanced from the previous, with the addition of the global variables. We model the prior for the centroid location as a Gaussian, i.e., $f_{\boldsymbol{\theta}}(\theta) = \mathcal{N}(\theta; \mu_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}})$. The phase variable $\boldsymbol{\phi}$ is assumed to take on a finite set of values $[1 \ldots n_{\boldsymbol{\phi}}]$ with probability mass function

$$f_{\boldsymbol{\phi}}(i) = \pi_{\boldsymbol{\phi},i} \quad i \in [1, \ldots, n_{\boldsymbol{\phi}}],$$

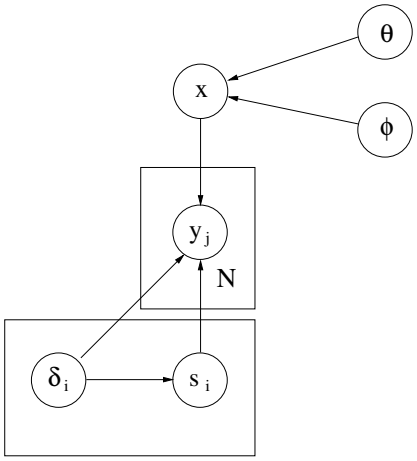


Figure 3.3: Global Variables. We complete the graphical model with the addition of two global variables which influence every body part. The centroid $\boldsymbol{\theta}$, is a continuous quantity indicating the "center of gravity" of the body as a whole. The phase $\boldsymbol{\phi}$ is a discrete variable used to index among a small set of "poses" or configurations.

where

$$\begin{aligned} \pi_{\boldsymbol{\phi},i} &\geq 0 \\ \sum_{i=1}^{n_{\boldsymbol{\phi}}} \pi_{\boldsymbol{\phi},i} &= 1. \end{aligned} \tag{3.16}$$

Since the centroid $\boldsymbol{\theta}$ and phase $\boldsymbol{\phi}$ are the parents-set of the body parts $\mathbf{x}$, we can write

$$f_{\boldsymbol{\theta}\boldsymbol{\phi}\mathbf{x}}(\theta, \phi, x) = f_{\boldsymbol{\theta}}(\theta) f_{\boldsymbol{\phi}}(\phi) f_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}(x|\theta, \phi).$$

Now, let's define

$$J = [\underbrace{\begin{array}{cccc} J_d & J_d & \cdots & J_d \end{array}}_{M}]^T \tag{3.17}$$

and

$$J_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In order to achieve translation invariance we model the collection of M body-parts as $x = J\theta + \overline{x}$, i.e., we hypothesize that a given pose x can be interpreted as a common displacement in position θ, superimposed to a translation-invariant or *centered* pose $\overline{x}$. We assume $\overline{\mathbf{x}}$ to be conditionally independent of $\boldsymbol{\theta}$ given $\boldsymbol{\phi}$, i.e., $f_{\overline{\mathbf{x}}|\boldsymbol{\theta}\boldsymbol{\phi}}(\overline{x}|\theta,\phi) = f_{\overline{\mathbf{x}}|\boldsymbol{\phi}}(\overline{x}|\phi)$, and we model its density as a linear Gaussian, that is,

$$f_{\overline{\mathbf{x}}|\boldsymbol{\phi}}(\overline{x}|\phi) = \mathcal{N}(\overline{x}; \mu_{\overline{\mathbf{x}}|\boldsymbol{\phi}}, \Sigma_{\overline{\mathbf{x}}|\boldsymbol{\phi}}). \tag{3.18}$$

Since the folllowing linear relationship holds

$$\begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} I & J \\ 0 & I \end{bmatrix} \begin{bmatrix} \overline{x} \\ \theta \end{bmatrix} \tag{3.19}$$

we conclude that

$$f_{\mathbf{x}\boldsymbol{\theta}|\boldsymbol{\phi}}(x,\theta|\phi) = \mathcal{N}\left(\begin{bmatrix} x \\ \theta \end{bmatrix}; \begin{bmatrix} \mu_{\overline{\mathbf{x}}|\boldsymbol{\phi}} + J\mu_{\boldsymbol{\theta}} \\ \mu_{\boldsymbol{\theta}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\overline{\mathbf{x}}|\boldsymbol{\phi}} + J\Sigma_{\boldsymbol{\theta}}J^T & J\Sigma_{\boldsymbol{\theta}} \\ \Sigma_{\boldsymbol{\theta}}J^T & \Sigma_{\boldsymbol{\theta}} \end{bmatrix} \right). \tag{3.20}$$

Recall from the previous sections that the level of independence among body parts $\mathbf{x}$ determines the size of the cliques in the graph. Furthermore, as noted in Section 3.2.2, the

computational complexity of our inference algorithms is dominated by the size of the largest clique in the graph. Now, since $\boldsymbol{\theta}$ belongs to the parent-set of every part $\mathbf{x}_i$, inference in the graphical model of Figure 3.3 is clearly intractable. This is the case even if we were to remove the discrete phase variable $\boldsymbol{\phi}$ from the set of global variables. Nevertheless, we notice that if we were given the true location of the centroid θ and the phase ϕ, the density of the body parts becomes

$$f_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}(x|\theta,\phi) = \mathcal{N}(x; \mu_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}, \Sigma_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}) \tag{3.21}$$

where

$$\begin{aligned} \mu_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}} &= \mu_{\overline{\mathbf{x}}|\boldsymbol{\phi}} + J\theta \\ \Sigma_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}} &= \Sigma_{\overline{\mathbf{x}}|\boldsymbol{\phi}}. \end{aligned}$$

Also, notice that since the structure of $\Sigma_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}$ is sparse by construction, observing the value of the global variables renders inference over the parts tractable again. Alternatively, if the labeling of the parts was known, we could easily determine the value of the global variables by maximizing their density, which reduces to a conditional linear Gaussian, or

$$f_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{x}_{\mathcal{F}}}(\theta,\phi|y_{s^f}) = \alpha(\phi, y_{s^f})\mathcal{N}(\theta; \mu_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{x}_{\mathcal{F}}}(\phi, y_{s^f}), \Sigma_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{x}_{\mathcal{F}}}(\phi, y_{s^f})) \tag{3.22}$$

where the normalization factor $\alpha(\phi, y_{s^f})$, the mean $\mu_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{x}_{\mathcal{F}}}(\phi, y_{s^f})$, and the convariance $\Sigma_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{x}_{\mathcal{F}}}(\phi, y_{s^f})$, are functions of both the unknown ϕ and the observed labeled data y_{s^f}.

Being able to answer either one of the two problems, given a solution for the other one, is encouraging since it suggests that *expectation maximization (EM)*, could be applicable.

In the following section we will present a derivation of EM for our problem, and then follow with an alternative optimization schema, based on Paskin's sample propagation work from [Pas03].

3.3.4.1 Expectation Maximization

Expectation maximization [DLR77] is a versatile algorithm which is often invoked when some of the variables in a model are not observable. In the problem of detection and labeling, we have already mentioned how it is convenient to assume the geometry of the body as a superimposition of a locally deformable model with an appropriate global translation within the frame. It is important to observe how this choice goes beyond making the interpretation of the model easy and intuitive. In fact, the introduction of hidden variables (and that of top-down modeling in general) is very often driven by our knowledge of the structure inherent to the problem; since we assume that a sparse level of dependency holds among the parts, we would like to preserve such structure in our model as much as it is possible, hence avoiding the devastating global coupling. The price to be paid to enjoy the benefit of isolating this global phenomenon amounts to a more involved inference process: should we choose otherwise, we would have to make all the parts interact, since any pair of parts would exhibit a strong correlation of their *absolute* location. This can be readily seen by observing that, at a macroscopic scale, the body moves as a whole and defines a bounded region which contains all its parts; knowing the position of one part tells us a lot about where all the others must lie.

The EM algorithm allows us to approach the problem by taking advantage of its inherent structure, while circumventing the issues due to the presence of hidden variables. The iterative process involves alternating between two steps which are called the *expectation*, or

E-Step, and the *maximization*, or *M-Step*.

In the E-Step, the best estimate for the hidden variables is computed, given the observed data. In the M-Step, maximum likelihood estimates for the parameters are computed, based on the data and the hidden variables. The process iterates until no more appreciable changes are shown by the model's likelihood.

One should notice that the choice of treating unknown quantities as hidden variables, or as parameters, is sometimes arbitrary and often driven by the convenience of computation. In our setting, we opt for treating the labeling variables $\mathbf{h}$ as hidden variables, while the global quantity $\boldsymbol{\theta}$ is considered a parameter.

We now proceed with the description of the E-Step and M-Step in more detail. To lighten the notation, and since we are only interested in translation invariance at this stage, we will ignore the phase variable $\boldsymbol{\phi}$.

E-Step

We start by marginalizing over $\mathbf{x}$, obtaining

$$\begin{aligned} f_{\mathbf{ys\delta\theta}}(y,s,\delta,\theta) &= \int f_{\mathbf{xys\delta\theta}}(x,y,s,\delta,\theta)dx \\ &= f_{\mathbf{s\delta}}(s,\delta)\int f_{\mathbf{x}|\boldsymbol{\theta}}(x|\theta)f_{\boldsymbol{\theta}}(\theta)\prod_{j=1}^{N} f_{\mathbf{y}_j|\mathbf{xs\delta}}(y_j|x,s,\delta)dx \\ &= f_{\mathbf{s\delta}}(s,\delta)\left(\frac{1}{V}\right)^{N-|\mathcal{F}|} f_{\boldsymbol{\theta}}(\theta)\int f_{\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}}(x_{\mathcal{F}}|\theta)\mathbf{1}\{y_{s^f}=x_{\mathcal{F}}\}dx_{\mathcal{F}} \\ &= \left(\frac{1}{V}\right)^{N-M} f_{\mathbf{s\delta}}(s,\delta)f_{\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}}(y_{s^f}|\theta)f_{\boldsymbol{\theta}}(\theta)\left(\frac{1}{V}\right)^{M-|\mathcal{F}|} \\ &\propto f_{\mathbf{x}_{\mathcal{F}}\boldsymbol{\theta}}(y_{s^f},\theta)\left(\frac{1}{V}\right)^{M-|\mathcal{F}|} \end{aligned} \tag{3.23}$$

where the last equation holds, since the prior over the labeling variables $\mathbf{h}=[\mathbf{s},\boldsymbol{\delta}]$ is uniform.

The *complete-data log-likelihood* becomes

$$\begin{aligned} L_c(y,h;\theta) &= \log\left[f_{\mathbf{yh\theta}}(y,h,\theta)\right] && (3.24)\\ &\propto \log\left[f_{\mathbf{x}_{\mathcal{F}}\boldsymbol{\theta}}(y_{s^f},\theta)\left(\frac{1}{V}\right)^{M-|\mathcal{F}|}\right]. && (3.25)\end{aligned}$$

As the hypothesis h is hidden, we are unable to maximize $L_c(y,h;\theta)$ directly and we have to replace it with the so called *expected-complete log-likelihood*, that is,

$$\hat{L}_c(\tilde{f},\theta) = E_{\tilde{f}_{\mathbf{h}}}[L_c(y,h;\theta)] \tag{3.26}$$

where the expectation is taken with respect to a generic distribution $\tilde{f}_{\mathbf{h}}(h)$.

Our goal is to find a density $f_{\mathbf{h}}$, and a value for θ, that maximizes (3.26) given the observed data y. Since this is an iterative process, we start with some initial estimate for the centroid, which we call $\theta^{(0)}$.

At iteration k, given $\theta^{(k-1)}$ as the current estimate for the value of the centroid, we would like to maximize $\hat{L}_c(\tilde{f},\theta)$ with respect to $\tilde{f}$. It can be shown (see, e.g., [DLR77]) that the optimal choice is

$$\tilde{f}^{(k)}_{\mathbf{h}}(h) = f_{\mathbf{h}|\mathbf{y}\boldsymbol{\theta}}(h|y,\theta^{(k-1)}). \tag{3.27}$$

This is not surprising since the conditional density (3.27) happens to be our "best guess" for the hypothesis h given all the information we have available from the observations y.

Unfortunately, we are unable to even represent (let alone compute) such density, since the number of possible assignments to h is exponential in the number of body parts. Once again we are forced to approximate. Instead of computing the full density, we make a

so-called *hard assignment*, i.e., we approximate $\tilde{f}_{\mathbf{h}}(h)$ with $\mathbf{1}(h - h^{(k)})$, where

$$h^{(k)} = \arg\max_{h} \left[f_{\mathbf{h}|\mathbf{y}\boldsymbol{\theta}}(h|y\theta^{(k-1)}) \right] \tag{3.28}$$

and $\mathbf{1}(z)$ is a Kronecker's delta centered in 0.

Now, we notice that

$$f_{\mathbf{h}|\mathbf{y}\boldsymbol{\theta}}(h|y,\theta^{(k-1)}) \quad \propto \quad f_{\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}}(y_{s^f}|\theta^{(k-1)}) \left(\frac{1}{V}\right)^{M-|\mathcal{F}|} \tag{3.29}$$

and since it factorizes, we can apply the max-prod algorithm on the loopy graph induced by the factorization, and solve the (3.28).

If the message-passing algorithm converges, and the determined $\tilde{f}^{(k)}$ maximizes the expected-complete log-likelihood $\hat{L}_c(\cdot, \theta^{(k-1)})$, we are guaranteed (otherwise there is just reasonable[3] hope.) that EM will converge to the sought-after ML estimate of $\boldsymbol{\theta}$.

M-Step

In the M-Step we compute the expectation of (3.26) using the current estimate $f_{\mathbf{h}}^{(k)}$, and we maximize it with respect to θ; that is, we compute

$$\begin{aligned} \theta^{(k+1)} &= \arg\max_{\theta} \hat{L}_c(\tilde{f}, \theta) \\ &= \arg\max_{\theta} \left[\log f_{\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}}(y_{s^f}|\theta) \right] \end{aligned}$$

[3] As we have already noted, experimentally it is observed that when LBP converges, the determined maximum is either global or, although local, the potential's value is very close to its global optimum. If the potential is increased (not necessarily maximized) by LBP, that suffices for EM to converge

where s^f and $\mathcal{F}$ have been set by the hypothesis $h^{(k)}$. The maximizing θ can be easily obtained from

$$0 = \nabla_\theta [f_{\mathbf{x}_\mathcal{F}|\boldsymbol{\theta}}(y_{s^f}|\theta)]. \tag{3.30}$$

Since the density is an un-normalized Gaussian potential in θ, whose moments are functions of h and y, the solution is a linear combination of the observations y.

3.3.4.2 Sample Propagation

As an alternative to the EM algorithm of the previous section, we propose to approach the maximization of the complete-data likelihood function more directly. The method consists of applying an efficient Markov chain Monte Carlo (MCMC) Gibbs sampler, based on Paskin's sample propagation work of [Pas03]. We now review their work and show how it can be applied to our optimization problem.

The Hybrid Model

Since this particular inference scheme will form the basis for our later work on movemes, we present it on the extended model that includes both the centroid $\boldsymbol{\theta}$ and the phase $\boldsymbol{\phi}$ among the global variables.

Let us consider the following conditional density, where we have already marginalized over the body parts $\mathbf{x}$:

$$f_{\mathbf{h}\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{y}}(h, \theta, \phi|y) \propto f_{\mathbf{x}_\mathcal{F}|\boldsymbol{\theta}\boldsymbol{\phi}}(y_{s^f}|\theta\phi) \left(\frac{1}{V}\right)^{M-|\mathcal{F}|} f_{\boldsymbol{\theta}}(\theta) f_{\boldsymbol{\phi}}(\phi) f_{\mathbf{h}}(h). \tag{3.31}$$

Given the observations y, we are interested in determining the most likely value for the centroid $\boldsymbol{\theta}$ and phase $\boldsymbol{\phi}$, as well as the labeling $\mathbf{h}$.

Another assumption we make is that a sparse level of dependency[4] holds among the parts $\mathbf{x}$, so that (3.31) factorizes as follows

$$f_{\mathbf{h}\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{y}}(h,\theta,\phi|y) = \frac{1}{Z(y)} \prod_{\mathcal{C}_i} \Psi_{\mathcal{C}_i}(h_{C_i},\theta,\phi,y), \tag{3.32}$$

where $Z(y)$ is the normalizing factor. (3.32) is the density we will work with for the reminder of this section.

In Section 3.3.2 we have mentioned how the problem of doing inference on a graphical model is in general that of computing the expectation of a function $U(\theta,\phi)$ defined over a subset of the variables, given the observations. That is,

$$\mathbb{E}_{f_{\mathbf{h}\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{y}}}\left[U(\theta,\phi)|y\right], \tag{3.33}$$

where $U(\theta,\phi)$ is a generic function of θ and ϕ. In particular, since we are interested in the centroid and phase themselves, we have that U is the identity function, i.e., $U(\theta,\phi) = [\theta,\phi]^T$. Notwithstanding the factorized form of (3.32), and the fact that the function of interest U only depends on a subset of the variables in the problem, we have already seen that the coupling induced by the global variables precludes the use of algorithms such as belief propagation.

Rao-Blackwellised Approximation

In an attempt to simplify the problem in (3.33), we notice that if we condition on the discrete variables $\mathbf{h}$, we are left with a conditional linear Gaussian (CLG) density in $\boldsymbol{\theta}$ and

[4] We assume that the graphical model among the body parts $\mathbf{x}$ has a decomposable triangulated structure. This is essentially the triangulated model of [SGP00].

$\boldsymbol{\phi}$, i.e.,

$$f_{\boldsymbol{\theta\phi}|\mathbf{hy}}(\theta, \phi|h, y) = \bar{\alpha}(\phi)\mathcal{N}\left(\theta; \bar{\mu}_\theta(\phi), \bar{\Sigma}_\theta(\phi)\right)$$

where we have omitted the dependence from h and y on the right-hand side. This makes it easy to compute the conditional expectation

$$\mathbb{E}_{f_{\boldsymbol{\theta\phi}|\mathbf{hy}}}\left[[\theta, \phi]^T|h, y\right]. \tag{3.34}$$

In fact, we can write

$$\begin{aligned}
\mathbb{E}_{f_{\boldsymbol{\theta\phi}|\mathbf{hy}}}\left[\begin{bmatrix}\theta\\ \phi\end{bmatrix}|h, y\right] &= \sum_\phi \int \begin{bmatrix}\theta\\ \phi\end{bmatrix} f_{\boldsymbol{\theta\phi}|\mathbf{hy}}(\theta, \phi|h, y)d\theta \\
&= \sum_\phi \int \begin{bmatrix}\theta\\ \phi\end{bmatrix} \bar{\alpha}(\phi)\mathcal{N}\left(\theta; \bar{\mu}_\theta(\phi), \bar{\Sigma}_\theta(\phi)\right) d\theta \\
&= \sum_\phi \begin{bmatrix}\bar{\mu}_\theta(\phi)\bar{\alpha}(\phi)\\ \phi\,\bar{\alpha}(\phi)\end{bmatrix}.
\end{aligned}$$

We then rewrite the (3.33) as

$$\mathbb{E}_{f_{\boldsymbol{\theta\phi}|\mathbf{y}}}\left[[\theta, \phi]^T|y\right] = \mathbb{E}_{f_{\mathbf{h}|\mathbf{y}}}\left[\mathbb{E}_{f_{\boldsymbol{\theta\phi}|\mathbf{hy}}}\left[[\theta, \phi]^T|h, y\right]|y\right] \tag{3.35}$$

where the outer expectation is, however, again intractable since the summation is over all possible hypothesis.

At this point we introduce a numerical approximation technique, called *Rao-Blackwellisation (R-B)*. An R-B approximation is achieved by extracting a reasonably large number of samples $\overline{h}^k$, $k = 1 \ldots K$ from the density of interest $f_{\mathbf{h}|\mathbf{y}}(\cdot|y)$. The outer summation in (3.35) is

then replaced with a tractable one, which yields the following estimator

$$\widehat{\mathcal{E}} \triangleq \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{f_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{hy}}}\left[[\theta,\phi]^T|\overline{h}^k, y\right]. \tag{3.36}$$

We then have

$$\mathbb{E}_{f_{\boldsymbol{\theta}\boldsymbol{\phi}|\mathbf{y}}}\left[[\theta,\phi]^T|y\right] \approx \widehat{\mathcal{E}} \tag{3.37}$$

with equality holding when we sample indefinitely.

Thanks to the R-B approximation, we have simplified the problem of computing the expectation in (3.34) to a tractable summation: we now show how to efficiently sample from the density $f_{\mathbf{h}|\mathbf{y}}(\cdot|y)$ by using a Blocking Gibbs sampler based on the junction tree algorithm.

Markov Chain Monte Carlo and Blocking Gibbs Sampling

Markov chain Monte Carlo (MCMC) methods generate samples from a complex distribution f by designing a Markov chain whose stationary distribution is f. The blocking Gibbs sampler is an example of MCMC where the next state of the Markov chain is chosen by resampling a subset of the variables, conditioned on the current values of the remaining variables.

Recall how the density (3.32) exhibits a factorization in families or clusters. This leads to an interesting way of applying blocking Gibbs sampling in our setting. Assume the current state of the Markov chain over h is $\overline{h}^k$. To generate the next state of the chain $\overline{h}^{k+1}$ we choose a clique (or cluster) C, and we resample h_C given $\overline{h}^k_{I\setminus C}$ from the following conditional distribution:

$$f_{\mathbf{h}_C|\mathbf{h}_{I\setminus C}\mathbf{y}}(\cdot|\overline{h}^k_{I\setminus C}, y). \tag{3.38}$$

The cluster C can be chosen randomly, or according to a schedule.

An Optimized Gibbs Sampler: Sample Propagation

The blocking Gibbs sampling and the Rao-Blackwellised estimation can be combined efficiently into a single algorithm. By passing *conditional* messages on a junction tree, the conditional densities (3.38) needed by the sampler can be computed efficiently. This is even more impressive if we think that the evidence is continuously being updated from one iteration to the next.

We start by defining, for each edge $B \longrightarrow C$, the following function over the variables θ, ϕ, $\mathbf{h}_{B\cap C}$, and the evidence on all the remaining labeling variables (which we indicate with $\overline{h}$)

$$\overline{m}_{BC}(\theta, \phi, h_{B\cap C}, \overline{h}) = \\ \Psi_B(\theta, \phi, h_{B\cap C}, \overline{h}_{B\setminus C}) \prod_{\substack{A\to B\in\mathcal{J} \\ A\neq C}} \overline{m}_{AB}(\theta, \phi, h_{A\cap B\cap C}, \overline{h}). \tag{3.39}$$

This is the *conditional message from B to C given the evidence $\overline{h}$*. Intuitively, when we send a message from B to C, we instantiate all evidence variables that are in B but not those that are in C. This gives us the freedom to later instantiate $\mathbf{h}_C$ as we wish.

We define the *conditional belief* on the cluster C as

$$\overline{b}_C(\theta, \phi, h_C, \overline{h}) = \Psi_C(\theta, \phi, h_C) \prod_{B\to C\in\mathcal{J}} \overline{m}_{BC}(\theta, \phi, h_{B\cap C}, \overline{h}). \tag{3.40}$$

For a given initial value of the evidence $\overline{h}^0$, we compute the conditional messages as in (3.39). We then compute the conditional cluster belief $\overline{b}_C(\theta, \phi, h_C, \overline{h}^0)$. Since we are

interested in the conditional distribution of h_C, we integrate out $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ from the belief. We then normalize it, obtaining

$$f_{\mathbf{h}_C|\mathbf{h}_{I\setminus C}\mathbf{y}}(h_C|\overline{h}^0_{I\setminus C}, y). \tag{3.41}$$

Next, we sample the conditional to obtain $\overline{h}^1_C$, which we use to update the evidence by setting $\overline{h}^1 = [\overline{h}^1_C, \overline{h}^0_{I\setminus C}]$.

With the new evidence we can go back to instantiate the belief $\overline{b}_C$, obtaining $\overline{b}_C(\theta, \phi, \overline{h}^1)$. Normalizing with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ finally yields

$$f_{\boldsymbol{\theta\phi}|\mathbf{hy}}(\theta, \phi|\overline{h}^1, y) \tag{3.42}$$

which is the density needed by the Rao-Blackwellised estimator. Every time we sample a new cluster C we are able to update the estimator, as $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ belong to every cluster in the junction tree.

The process continues by

- jumping to the next cluster according to some schedule,
- computing the conditional on the junction tree, which we use in the Gibbs sampler, and
- using the newly produced sample to generate the conditional over $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$; this is accumulated by the estimator.

We notice that when we resample a cluster C_p, we have $\overline{h}^k_{[I\setminus C_p]} = [\overline{h}^{k-1}_{[I\setminus C_p]}]$; that is, the evidence variables that are not in C_p are unchanged. In [Pas03] it is shown how this condition, and the junction property, imply that when we resample the variables in C_p the

messages directed towards C_p do not change and need not be recomputed. On the other hand, if C_q is the next cluster we sample, only the messages directed towards C_q are needed by the definitions (3.39) and (3.40).

Combining the two arguments, [Pas03] shows that *only the messages on the directed path from C_p to C_q must be recomputed in iteration k+1.* Since we have the freedom to select a specific schedule for visiting the clusters, we can achieve a noticeable computational saving. In fact, if we choose the order such that C_q is always a neighbor of C_p, we only have to recompute a single message (from C_p to C_q) at each iteration.

Having seen how to use sample propagation to compute estimates of both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, we observe that a large number of high-probability samples for the labeling variables $\mathbf{h}$ are obtained as a by-product. By retaining the most likely one in the model we complete our task of labeling the human being.

3.4 Learning

In the learning process we want to estimate the structure and parameters of $f_{\mathbf{x}}(x)$ from the data.

In the most common scenario, we are given a training dataset of observations y, where the identity or labeling $\mathbf{s}$ of the data in each frame is known. Additionally, its convenient to craft the dataset so that $N = M$; that is no clutter is present. Furthermore, we assume that all parts are visible in every frame.

For the sake of simplicity, let us initially assume that the moving human is centrally positioned in the frame, so that we do not have to account for horizontal and vertical translations in the data. Since the data we use is labeled, we can easily center it.

We assume each body part x_i to be a d dimensional vector and their joint distribution

$f_{\mathbf{x}}(x)$ to be Gaussian $\mathcal{N}(\mu, \Sigma)$. Here μ and Σ are the sample mean and covariance obtained from the labeled data y_{sf}.

3.4.1 Structure Learning

Learning the structure of the model is a challenging task. As in [SGP00], we adopt the approximation of limiting the number of dependencies among parts (i.e., the fan-in of each node in the graph) to a fixed value $K = 2$. In contrast to Song, however, we dispose of the decomposability requirement and allow for a more general structure of dependencies.

We begin by introducing a scoring method, for a model, which is known as the *minimum description length (MDL)*, the *Bayesian information criteria (BIC)* or the *Schwartz information criteria*. The MDL/BIC principle is widely used in statistics as a model selection tool and offers interesting asymptotic properties (see [Sch78], [LB94], [BJ02]). The idea behind MDL is to take into consideration both the "goodness of fit" of a model to the data, as well as the complexity of the model itself. As we have already observed, a fully connected graphical model would be the most accurate description of the training set, yet the least useful, since a search for the optimal labelling would be computationally infeasible. Additionally, by Occam's razor, if the goodness of fit was the same, a more complex model might not generalize as well as a simpler one.

To quantify this trade off, the MDL/BIC principle suggests scoring a model from an information theory point of view. An arc in the graph indicates a dependence among two vertices. If we need to estimate the value of the dependent variable, then knowing the value of its parents provides us (on average) information; that is, we have less uncertainty about the child and thus need less bits to convey its value. The stronger the child-parents dependence, the fewer bits are needed.

The average amount of additional information on the child, provided by observing the parents, is exactly what the mutual information $\mathcal{I}(i, \pi_i)$ of a family represents. An alternative interpretation is that the mutual information reflects the likelihood that the data satisfies the dependency relationship.

On the other hand, representing this relationship incurs a cost. Imagine if we had to transmit the model over a channel: the higher the number of connections, the larger the number of bits required for its transmission.

The MDL/BIC principle combines these two quantities into a single score. In the case of Gaussian models, it is easy to see that the mutual information is given by a ratio of determinants, while the cost of representing the model is proportional to the number of non-zero entries in the inverse of the covariance matrix. For N i.i.d. data, and a family (i, π_i), we have

$$BIC(i, \pi_i) = \frac{N}{2} \log_2 \frac{|\Sigma_{i\cup\pi_i, i\cup\pi_i}|}{|\Sigma_{\pi_i,\pi_i}||\Sigma_{i,i}|} + \frac{d_{\pi_i} d}{2} \log_2 N \ , \qquad d_{\pi_i} \triangleq |\pi_i| d$$

while the total score of the graph $\mathcal{G}$ is

$$BIC(\mathcal{G}) = \sum_{i=1}^{M} BIC(i, \pi_i). \tag{3.43}$$

Ideally, we would like to examine every possible graphical model that can be constructed over the M variables in our problem, and score each one using the metric (3.43). For each of the graphs we could evaluate the encoding length of the data and that of the model description, searching for the one that minimizes their sum. However, this method is clearly impractical since there is an exponential number of graphs over the M variables. Unfortunately, the problem of finding the optimal graph has been shown to be NP-hard [Chi96].

A number of heuristics could be applied to the problem. We choose the solution of Giudici et al. [GC03], which is based on the Markov chain Monte Carlo (MCMC) method. Sampling methods provide an excellent tool for hard optimization problems and their empirical performance is well documented.

The algorithm is initialized by arbitrarily choosing a feasible graph. At every iteration a move is proposed at random by choosing among three possibilities:

- **Addition:** a new arc is added between randomly chosen variables, as long as structural constrains such as maximum fan-in and the absence of cycles are maintained.

- **Deletion:** an existing arc is removed from the graph.

- **Reversal:** the direction of an existing arc is switched, subject to the same constraints imposed on additions.

The graph obtained after the proposed move is evaluated according to equation (3.43) and its score compared to that of the existing graph. Acceptance of the move is guaranteed only if the score is increased. If, on the other hand, the newly obtained graph fares less than the current one, a probability of acceptance is computed that is lower as the decrease in score contributed by the move grows. A key aspect in avoiding local minima is that MCMC methods accept (with appropriate reluctance) moves that decrease the score of the functional being optimized. Although this seems counter-intuitive, it is important to notice how this creates an escape from local attraction basins, allowing the exploration of larger portions of the solution space. An additional step we take, trying to mitigate the curse of local minima, is to randomly restart the algorithm several times, and retain the best performing graph of all runs.

3.4.2 Unsupervised Learning: Expectation Maximization

In the previous section we have assumed that the input to our learning algorithm was made of *labeled* and *complete* data. That is, on any given frame, all M parts were assumed to have been detected and their identity known. To obtain such a complete input, one can proceed in either one of two ways:

Motion Capture: An actor performs a few activities of interest within an arena, while wearing a special suit or tight clothing. A number of reflective markers, attached to the joints of the body, are tracked by several cameras located around the arena. The identity and position of each marker in each frame is established by combining the different video feeds.

Hand Labeling: A less intrusive way of producing the training set is to take a (possibly already existing) video and go through it frame by frame, while manually selecting the locations of the joints (or other relevant part).

Although, both methods have been extensively used, they both suffer from a number of drawbacks. In one case, the major issue is with the constraints imposed by the motion-capture system. The recording area is generally limited and the equipment cannot be easily transported. Also, the special clothing is often problematic if an unmodified video stream of the same actions is also needed (e.g., if the system being developed could in principle use both motion and appearance as inputs). The hand-labeling process, on the other hand, involves a great amount of human labor and patience, which is normally beyond what we are willing to invest in order to produce a sufficiently large dataset.

For the reasons just mentioned, it is strongly appealing to be able to learn from unlabeled and possibly incomplete data. It turns out that, although challenging, it is not an impossible

task.

In Section 3.3.4.1 we have seen an application of the EM algorithm to the labeling problem. There, the solution involved alternatively optimizing with respect to the labeling variables and the global quantities. A similar situation can be observed here. If we knew the parameters and structure of the model, we could use one of the inference techniques presented to label the data. Then, since the data would now be labeled, we could improve the model by re-estimating its parameters and structure. Once again, the EM algorithm seems to have solved our dilemma. It is important to note, however, that we have no guarantee on the completeness of the data after we label it, since parts could be missing, or the inference procedure could fail from time to time. Luckily, this can be taken into account when computing expectations in the E-step of the algorithm, as we will see shortly.

We start by defining the complete-data likelihood function

$$\begin{aligned} L_c(\theta, \phi, x, y, h|G) &= \log \prod_{t=1}^{T} f_{\boldsymbol{\theta\phi xyh}|\mathbf{G}}(\theta^t, \phi^t, x^t, y^t, h^t|G) \\ &= \sum_{t=1}^{T} \log f_{\boldsymbol{\theta\phi xyh}|\mathbf{G}}(\theta^t, \phi^t, x^t, y^t, h^t|G). \end{aligned}$$

Here, y is the set of observations; the labeling variables $\mathbf{h}$, the centroid $\boldsymbol{\theta}$, the phase $\boldsymbol{\phi}$, and the geometry of the parts $\mathbf{x}$ are considered hidden variables, while the structure and parameters G of the graphical model $\mathcal{G}$ are the quantities we wish to estimate. We use the superscript $t \in [1 \dots T]$ as a time index; for example, y^t indicates the collection of observations in the t-th frame (and, similarly, for the other variables).

As introductions of the general framework for the EM algorithm are widely available in the literature (see [Bis95], [Bis07], or [Jor], among others), we will only briefly mention the iterative nature of the scheme, and focus instead on the derivation of the equations for the

k-th iteration of E-step and M-step, as they apply to the specific of our problem.

Expected-Complete Log-Likelihood

Since the variables $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\mathbf{x}$, and $\mathbf{h}$ can not be observed, we compute the expected-complete log-likelihood with respect to the generic distribution $q(\theta, \phi, x, h)$, obtaining

$$\begin{aligned} L_e(y|G,q) &= \mathbb{E}_q\left[L_c(\theta,\phi,x,y,h|G)\right] \\ &= \sum_{\phi,h}\int q(\theta,\phi,x,h)\sum_{t=1}^{T}\log f_{\boldsymbol{\theta\phi xyh|G}}(\theta^t,\phi^t,x^t,y^t,h^t|G)dxd\theta. \end{aligned} \quad (3.44)$$

E step

In the E-Step we maximize the (3.44) with respect to the averaging density $q(\theta, \phi, x, h)$, while holding the model parameters G fixed. It has been proved that the maximizing choice is given by the conditional a posteriori density of the hidden variables, given the data

$$q^{(k+1)}(\theta,\phi,x,h) = f_{\boldsymbol{\theta\phi xh|yG}}(\theta,\phi,x,h|y,G^{(k)}).$$

M step

In the M-Step we take the expected complete log-likelihood (where the expectation is computed with respect to the distribution determined in the E-Step) and maximize it with respect to the model parameters G, yielding the improved model

$$G^{(k+1)} = \arg\max_{G}\left[L_e(y|G,q^{(k+1)})\right].$$

Now that we have described the general structure of the algorithm, let us proceed with the derivation of both steps.

We start by noticing that the summation over t and the (linear) expectation operator can be swapped, yielding

$$\begin{aligned} L_e(y|G,q) &= \mathbb{E}_q\left[L_c(\theta,\phi,x,y,h|G)\right] \\ &= \mathbb{E}_q\left[\sum_{t=1}^{T}\log f_{\boldsymbol{\theta\phi xyh}|\mathbf{G}}(\theta^t,\phi^t,x^t,y^t,h^t|G)\right] \\ &= \sum_{t=1}^{T}\mathbb{E}_q\left[\log f_{\boldsymbol{\theta\phi xyh}|\mathbf{G}}(\theta^t,\phi^t,x^t,y^t,h^t|G)\right]. \end{aligned}$$

Now, the logarithm transforms the factorization into a summation, and allows us to decouple the maximization into individual terms. As noted in [Wel00], the fact that $f_{\mathbf{y}|\mathbf{xhG}}(\cdot|\cdot,\cdot,G)$ is not a conventional function but rather a Dirac's delta can be disturbing when computing its logarithm. However, since none of the parameters with respect to which we are optimizing is contained in such a term, we simply omit that term from each expectation, and write the following

$$\begin{aligned} &\mathbb{E}_q\left[\log f_{\boldsymbol{\theta\phi xyh}|\mathbf{G}}(\theta^t,\phi^t,x^t,y^t,h^t|G)\right] \propto \ldots \\ &\quad \mathbb{E}_q\left[\log f_{\boldsymbol{\theta}|\mathbf{G}}(\theta^t|G)\right] + \mathbb{E}_q\left[\log f_{\boldsymbol{\phi}|\mathbf{G}}(\phi^t|G)\right] + \ldots \\ &\quad \mathbb{E}_q\left[\log f_{\mathbf{x}|\boldsymbol{\theta\phi}\mathbf{G}}(x^t|\theta^t,\phi^t,G)\right] + \ldots \\ &\quad \mathbb{E}_q\left[\log f_{\mathbf{s}|\boldsymbol{\delta}\mathbf{G}}(s^t|\delta^t,G)\right] + \mathbb{E}_q\left[\log f_{\boldsymbol{\delta}|\mathbf{G}}(\delta^t|G)\right] \end{aligned}$$

where the "$\propto$" means that maximizing either side with respect to the distribution's parameters yields the same result. We now rewrite each term, replacing the logarithm of conditional factors with their actual expressions, i.e.,

Prior on the Centroid :

$$\mathbb{E}_q\big[\log f_{\boldsymbol{\theta}|\mathbf{G}}(\theta^t|G)\big] =$$
$$= \mathbb{E}_q\Big[-\frac{1}{2}\log|2\pi\Sigma_{\boldsymbol{\theta}}| - \frac{1}{2}\big[\theta^t - \mu_{\boldsymbol{\theta}}\big]^T\Sigma_{\boldsymbol{\theta}}^{-1}\big[\theta^t - \mu_{\boldsymbol{\theta}}\big]\Big]$$

Prior on the Phase :

$$\mathbb{E}_q\big[\log f_{\boldsymbol{\phi}|\mathbf{G}}(\phi^t|G)\big] = \mathbb{E}_q\big[\log(\pi_{\boldsymbol{\phi},\phi^t})\big]$$

Prior on the Labeling Variables :

$$\mathbb{E}_q\big[\log f_{\mathbf{s}|\boldsymbol{\delta}\mathbf{G}}(s^t|\delta^t, G)\big] = \mathbb{E}_q\big[-M\log N\big]$$
$$\mathbb{E}_q\big[\log f_{\boldsymbol{\delta}|\mathbf{G}}(\delta^t|G)\big] = \mathbb{E}_q\left[\sum_{i=1}^{M}\big[\delta_i^t\log p_i + (1-\delta_i^t)\log(1-p_i)\big]\right]$$

Density on the Body Parts :

$$\mathbb{E}_q\big[\log f_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}\mathbf{G}}(x^t|\theta^t,\phi^t,G)\big] =$$
$$= \mathbb{E}_q\Bigg[-\frac{1}{2}\log|2\pi\Sigma_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}(\theta^t,\phi^t)|$$
$$-\frac{1}{2}\big[x^t - \mu_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}(\theta^t,\phi^t)\big]^T\Sigma_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}^{-1}(\theta^t,\phi^t)\big[x^t - \mu_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}}(\theta^t,\phi^t)\big]\Bigg]$$
$$= \mathbb{E}_q\Bigg[-\frac{1}{2}\log|2\pi\Sigma_{x|\boldsymbol{\phi}}(\phi^t)|$$
$$-\frac{1}{2}\big[x^t - \mu_{x|\boldsymbol{\phi}}(\phi^t) - J\theta^t\big]^T\Sigma_{x|\boldsymbol{\phi}}^{-1}(,\phi^t)\big[x^t - \mu_{x|\boldsymbol{\phi}}(\phi^t) - J\theta^t\big]\Bigg].$$

We now proceed to compute the maximum likelihood parameters by setting to zero the partial derivatives of the likelihood with respect to the parameters of G.

Prior on the Centroid :

Let us first take the derivative with respect to $\mu_{\boldsymbol{\theta}}$:

$$0 = \frac{\partial L_e(y, b|G, q^{(k+1)})}{\partial \mu_{\boldsymbol{\theta}}} = \sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}} \left[\Sigma_{\boldsymbol{\theta}}^{-1} \left(\theta^t - \mu_{\boldsymbol{\theta}} \right) \right]$$

$$\Longrightarrow \qquad \mu_{\boldsymbol{\theta}}^{(k+1)} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}} \left[\theta^t \right].$$

We then take the derivative with respect to $\Sigma_{\boldsymbol{\theta}}^{-1}$ obtaining:

$$\begin{aligned} 0 &= \frac{\partial L_e(y, b|G, q^{(k+1)})}{\partial \Sigma_{\boldsymbol{\theta}}^{-1}} \\ &= \frac{1}{2} \sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}} \left[\Sigma_{\boldsymbol{\theta}} - \left(\theta^t - \mu_{\boldsymbol{\theta}} \right) \left(\theta^t - \mu_{\boldsymbol{\theta}} \right)^T \right] \end{aligned}$$

$$\Longrightarrow \qquad \Sigma_{\boldsymbol{\theta}}^{(k+1)} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}} \left[\theta^t \theta^{t^T} \right] - \mu_{\boldsymbol{\theta}}^{(k+1)} \mu_{\boldsymbol{\theta}}^{(k+1)^T}.$$

Prior on the Phase :

For $i \in [1, \ldots, n_{\boldsymbol{\phi}}]$ we maximize L_e with respect to $\pi_{\boldsymbol{\phi},i}$. Since we have a normalization constraint to accommodate, we introduce a Lagrange multiplier λ, obtaining:

$$\begin{aligned} 0 &= \frac{\partial}{\partial \pi_{\boldsymbol{\phi},i}} \left[L_e(y, b|G, q^{(k+1)}) + \lambda \left(\sum_{i=1}^{n_{\boldsymbol{\phi}}} \pi_{\boldsymbol{\phi},i} - 1 \right) \right] \\ &= \frac{\partial}{\partial \pi_{\boldsymbol{\phi},i}} \left[\sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}} \left[\log \pi_{\boldsymbol{\phi},\phi^t} \right] + \lambda \left(\sum_{i=1}^{n_{\boldsymbol{\phi}}} \pi_{\boldsymbol{\phi},i} - 1 \right) \right] \\ &= \frac{T_i^{\boldsymbol{\phi}}}{\pi_{\boldsymbol{\phi},i}} + \lambda_2 \end{aligned}$$

or

$$\pi_{\boldsymbol{\phi},i} \;=\; \frac{T_i^{\boldsymbol{\phi}}}{\lambda}$$

where $T_i^{\boldsymbol{\phi}} = \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{ybG}}(i|y, G^{(k)})$. Summing over i on both sides of the equation, and enforcing the normalization constraint, we get

$$\lambda = \sum_{i=0}^{n_{\boldsymbol{\phi}}-1} T_i^{\boldsymbol{\phi}} \triangleq T$$

$$\Longrightarrow \qquad \pi_{\boldsymbol{\phi},i} = \frac{T_i^{\boldsymbol{\phi}}}{T}.$$

Prior over the Labeling Variables :

Since the prior over the labeling variables $\mathbf{s}$ is set to a constant value, we are done with it and proceed to maximize with respect to the probability of detections. We set the derivative with respect to p_i to zero, obtaining:

$$\begin{aligned}
0 \;&=\; \frac{\partial L_e(y|G, q^{(k+1)})}{\partial p_i} \\
&=\; \frac{\partial}{\partial p_i}\left[\sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}}\left[\sum_{i=1}^{M}\left[\delta_i^t \log p_i + (1-\delta_i^t)\log(1-p_i)\right]\right]\right] \\
&=\; \sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}}\left[\frac{\delta_i^t}{p_i} - \frac{1-\delta_i^t}{1-p_i}\right] \\
&=\; \sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}}\left[\frac{\delta_i^t - p_i}{(1-p_i)p_i}\right].
\end{aligned}$$

Therefore,

$$\Longrightarrow \qquad p_i^{(k+1)} \;=\; \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}}\left[\delta_i^t\right].$$

Density of the Body Parts :

We compute the derivative with respect to $\mu_{x|\boldsymbol{\phi}}(j)$ for $j \in [1, \ldots, n_{\boldsymbol{\phi}}]$, obtaining:

$$
\begin{aligned}
0 &= \frac{\partial L_e(y, b|G, q^{(k+1)})}{\partial \mu_{x|\boldsymbol{\phi}}(j)} \\
&= \frac{\partial}{\partial \mu_{x|\boldsymbol{\phi}}(j)} \left[\sum_{t=1}^{T} \mathbb{E}_{q^{(k+1)}} \left[\log f_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}\mathbf{G}}(x^t|\theta^t, \phi^t, G) \right] \right] \\
&= \frac{\partial}{\partial \mu_{x|\boldsymbol{\phi}}(j)} \left[\sum_{t=1}^{T} \mathbb{E}_{f_{\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{x}^t\mathbf{h}^t|\mathbf{yG}}(\theta^t, \phi^t, x^t, h^t|y, G^{(k)})} \left[\cdots \right] \right] \\
&= \frac{\partial}{\partial \mu_{x|\boldsymbol{\phi}}(j)} \left[\sum_{t=1}^{T} \mathbb{E}_{f_{\boldsymbol{\phi}^t|\mathbf{yG}}(\phi^t|y, G^{(k)})} \left[\mathbb{E}_{f_{\boldsymbol{\theta}^t\mathbf{x}^t\mathbf{h}^t|\boldsymbol{\phi}^t\mathbf{yG}}(\theta^t, x^t, h^t|\phi^t, y, G^{(k)})} \left[\cdots \right] \right] \right] \\
&= \frac{\partial}{\partial \mu_{x|\boldsymbol{\phi}}(j)} \left[\sum_{t=1}^{T} \left[f_{\boldsymbol{\phi}^t|\mathbf{ybG}}(j|y, G^{(k)}) \mathbb{E} \left[\log f_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}\mathbf{G}}(x^t|\theta^t, j, G) \right] \right] \right] \\
&= \sum_{t=1}^{T} \left[f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f\ldots} \left[\Sigma_{x|\boldsymbol{\phi}}^{-1}(j) \left[x^t - \mu_{x|\boldsymbol{\phi}}(j) - J\theta^t \right] \right] \right] \\
&= \Sigma_{x|\boldsymbol{\phi}}^{-1}(j) \sum_{t=1}^{T} \left[f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f\ldots} \left[\left[x^t - \mu_{x|\boldsymbol{\phi}}(j) - J\theta^t \right] \right] \right] \\
&= \Sigma_{x|\boldsymbol{\phi}}^{-1}(j) \left[-T_j^{\phi} \mu_{x|\boldsymbol{\phi}}(j) + \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f\ldots} \left[(x^t - J\theta^t) \right] \right]
\end{aligned}
$$

where we defined $T_j^{\phi} = \sum_{t=1}^{T} f_{\ \boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)})$. We finally have:

$$
\begin{aligned}
&\mu_{x|\boldsymbol{\phi}}^{(k+1)}(j) = \\
&= \frac{1}{T_j^{\phi}} \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f_{\boldsymbol{\theta}^t\mathbf{x}^t\mathbf{h}^t|\boldsymbol{\phi}^t\mathbf{yG}}(\theta^t, x^t, h^t|j, y, G^{(k)})} \left[(x^t - J\theta^t) \right].
\end{aligned}
$$

Proceeding in a similar way we compute the derivatives with respect to $\Sigma_{x|\boldsymbol{\phi}}^{-1}(j)$. For

$j \in [1, \ldots, n_{\boldsymbol{\phi}}]$ we get:

$$
\begin{aligned}
0 &= \frac{\partial L_e(y|G, q^{(k+1)})}{\partial \Sigma^{-1}_{x|\boldsymbol{\phi}}(j)} \\
&\vdots \\
&= \frac{\partial}{\partial \Sigma^{-1}_{x|\boldsymbol{\phi}}(j)} \Bigg[-\frac{1}{2} \sum_{t=1}^{T} \Bigg[f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \cdots \\
&\quad \cdots \mathbb{E}_{f_{\ldots|\boldsymbol{\phi}^t \ldots}(\ldots|j,\ldots)} \left[\log f_{\mathbf{x}|\boldsymbol{\theta}\boldsymbol{\phi}\mathbf{G}}(x^t|\theta^t, j, G) \right] \Bigg] \Bigg] \\
&= \frac{1}{2} \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f_{\boldsymbol{\theta}^t \mathbf{x}^t \ldots}} \Big[\Sigma^{-1}_{x|\boldsymbol{\phi}}(j) \cdots \\
&\quad \cdots - \left[x^t - \mu_{x|\boldsymbol{\phi}}(j) - J\theta^t \right] \left[x^t - \mu_{x|\boldsymbol{\phi}}(j) - J\theta^t \right]^T \Big] \\
&= \frac{1}{2} T^{\boldsymbol{\phi}}_j \left[\Sigma^{-1}_{x|\boldsymbol{\phi}}(j) - \mu_{x|\boldsymbol{\phi}}(j) \mu_{x|\boldsymbol{\phi}}(j)^T \right] \cdots \\
&\quad \cdots - \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f_{\boldsymbol{\theta}^t \mathbf{x}^t \ldots}} \left[\left[x^t - J\theta^t \right] \left[x^t - J\theta^t \right]^T \right]
\end{aligned}
$$

where, again, we defined $T^{\boldsymbol{\phi}}_j = \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)})$. We then have:

$$
\begin{aligned}
\Longrightarrow \qquad & \Sigma^{(k+1)}_{x|\boldsymbol{\phi}}(j) = -\mu^{(k+1)}_{x|\boldsymbol{\phi}}(j) \mu^{(k+1)}_{x|\boldsymbol{\phi}}(j)^T + \cdots \\
& \cdots \frac{1}{T^{\boldsymbol{\phi}}_j} \sum_{t=1}^{T} f_{\boldsymbol{\phi}^t|\mathbf{yG}}(j|y, G^{(k)}) \mathbb{E}_{f_{\boldsymbol{\theta}^t \mathbf{x}^t \ldots}} \left[\left[x^t - J\theta^t \right] \left[x^t - J\theta^t \right]^T \right].
\end{aligned}
$$

So far we have reduced the problem of maximizing the expected-complete likelihood to that of computing a few (conditional) expectations of one or two variables. However, the densities involved in the integrals/summations are still intractable and we are forced to approximate them. More precisely, we will replace the densities of $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\mathbf{s}$, and $\boldsymbol{\delta}$ with Dirac's (or Kronecker's) deltas centered on their maximum a posteriori (MAP) values.

We start from the expectation of $\boldsymbol{\theta}$. We have

$$\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[\theta^t\right] &= \int \theta^t f_{\boldsymbol{\theta}^t|\mathbf{yG}}(\theta^t|y, G^{(k)})d\theta^t \\
&= \int \theta^t \mathbf{1}\{\theta^t = \hat{\theta}^t\}d\theta^t \\
&= \hat{\theta}^t
\end{aligned}$$

where the hat denotes the MAP estimate for the variable. Similarly we have,

$$\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[\theta^t\theta^{t^T}\right] &= \int \theta^t\theta^{t^T} f_{\boldsymbol{\theta}^t|\mathbf{yG}}(\theta^t|y, G^{(k)})d\theta^t \\
&= \int \theta^t\theta^{t^T} \mathbf{1}\{\theta^t = \hat{\theta}^t\}d\theta^t \\
&= \hat{\theta}^t\hat{\theta}^{t^T}.
\end{aligned}$$

For the labeling variables we have

$$\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[\delta_i^t\right] &= \sum_{\delta^t=0}^{1} \delta^t f_{\boldsymbol{\delta}^t|\mathbf{yG}}(\delta^t|y, G^{(k)}) \\
&= \sum_{\delta^t=0}^{1} \delta^t \mathbf{1}\{\delta^t = \hat{\delta}^t\} \\
&= \hat{\delta}^t.
\end{aligned}$$

The expectations for the sufficient statistics of $\mathbf{x}$ are slightly more challenging. First, we derive a convenient expression for $q^{(k+1)}$:

$$
\begin{aligned}
q^{(k+1)}(\theta^t, \phi^t, x^t, h^t) = & \\
&= f_{\boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{x}^t \mathbf{h}^t | \mathbf{y}^t \mathbf{G}}(\theta^t, \phi^t, x^t, h^t | y^t, G^{(k)}) \\
&= f_{\mathbf{x}^t | \mathbf{y}^t \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{h}^t \mathbf{G}}(x^t | y^t, \theta^t, \phi^t, h^t, G^{(k)}) f_{\boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{h}^t | \mathbf{y}^t \mathbf{G}}(\theta^t, \phi^t, h^t | y^t, G^{(k)}).
\end{aligned}
$$

The first factor on the right-hand side can be rewritten as

$$
\begin{aligned}
f_{\mathbf{x}^t | \mathbf{y}^t \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{h}^t \mathbf{G}}(x^t | y^t, \theta^t, \phi^t, h^t, G^{(k)}) = & \\
&= \frac{f_{\mathbf{y}^t | \mathbf{x}^t \mathbf{h}^t \mathbf{G}}(y^t | x^t, h^t, G^{(k)}) f_{\mathbf{x}^t | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{G}}(x^t | \theta^t, \phi^t, G^{(k)})}{f_{\mathbf{y}^t | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{h}^t \mathbf{G}}(y^t | \theta^t, \phi^t, h^t, G^{(k)})} \\
&= \frac{f_{\mathbf{y}^t | \mathbf{x}^t \mathbf{h}^t \mathbf{G}}(y^t | x^t, h^t, G^{(k)}) f_{\mathbf{x}^t | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{G}}(x^t | \theta^t, \phi^t, G^{(k)})}{\int f_{\mathbf{y}^t | \mathbf{x}^t \mathbf{h}^t \mathbf{G}}(y^t | x^t, h^t, G^{(k)}) f_{\mathbf{x}^t | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{G}}(x^t | \theta^t, \phi^t, G^{(k)}) dx} \\
&= \frac{f_{\mathbf{y}^t | \mathbf{x}^t \mathbf{h}^t \mathbf{G}}(y^t | x^t, h^t, G^{(k)}) f_{\mathbf{x}^t | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{G}}(x^t | \theta^t, \phi^t, G^{(k)})}{f_{\mathbf{x}^t_{\mathcal{F}} | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{h}^t \mathbf{G}}(y^t_{s_{\mathcal{F}}} | \theta^t, \phi^t, h^t, G^{(k)}) \left(\frac{1}{V}\right)^{N - |\mathcal{F}|}}
\end{aligned}
$$

where the subscript $s_{\mathcal{F}}$ in $y_{s_{\mathcal{F}}}$ is the MAP estimate, but we omitted the "hat".

The expectation becomes

$$
\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[x^t\right] &= \int x^t f_{\boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{x}^t \mathbf{h}^t | \mathbf{y}^t \mathbf{G}}(\theta^t, \phi^t, x^t, h^t | y^t, G^{(k)}) d\theta^t d\phi^t dx^t dh^t \\
&= \int x^t \frac{f_{\mathbf{y}^t | \mathbf{x}^t \mathbf{h}^t \mathbf{G}}(y^t | x^t, h^t, G^{(k)}) f_{\mathbf{x}^t | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{G}}(x^t | \theta^t, \phi^t, G^{(k)})}{f_{\mathbf{x}^t_{\mathcal{F}} | \boldsymbol{\theta}^t \boldsymbol{\phi}^t \mathbf{h}^t \mathbf{G}}(y^t_{s_{\mathcal{F}}} | \theta^t, \phi^t, h^t, G^{(k)}) \left(\frac{1}{V}\right)^{N - |\mathcal{F}|}} dx^t.
\end{aligned}
$$

We now split the vector x^t into its foreground and missing components ($x^t_{\mathcal{F}}$ and $x^t_{\mathcal{M}}$), and compute the expectation separately.

$$\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{F}}\right] &= \int x^t_{\mathcal{F}} \frac{f_{\mathbf{y}^t|\mathbf{x}^t\mathbf{h}^t\mathbf{G}}(y^t|x^t,h^t,G^{(k)}) f_{\mathbf{x}^t|\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{G}}(x^t|\theta^t,\phi^t,G^{(k)})}{f_{\mathbf{x}^t_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{h}^t\mathbf{G}}(y^t_{s_F}|\theta^t,\phi^t,h^t,G^{(k)}) \left(\frac{1}{V}\right)^{N-|\mathcal{F}|}} dx^t \\
&= y^t_{s_{\mathcal{F}}} \frac{f_{\mathbf{x}^t_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{h}^t\mathbf{G}}(y^t_{s_{\mathcal{F}}}|\theta^t,\phi^t,h^t,G^{(k)}) \left(\frac{1}{V}\right)^{N-|\mathcal{F}|}}{f_{\mathbf{x}^t_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{h}^t\mathbf{G}}(y^t_{s_{\mathcal{F}}}|\theta^t,\phi^t,h^t,G^{(k)}) \left(\frac{1}{V}\right)^{N-|\mathcal{F}|}} \\
&= y^t_{s_{\mathcal{F}}}
\end{aligned}$$

$$\begin{aligned}
&\mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{M}}\right] = \\
&= \int x^t_{\mathcal{M}} \frac{f_{\ldots}(y^t|x^t,h^t,G^{(k)}) f_{\ldots}(x^t_{\mathcal{M}}|x^t_{\mathcal{F}},\theta^t,\phi^t,G^{(k)}) f_{\ldots}(x^t_{\mathcal{F}}|,\theta^t,\phi^t,G^{(k)})}{f_{\mathbf{x}^t_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{h}^t\mathbf{G}}(y^t_{s_{\mathcal{F}}}|\theta^t,\phi^t,h^t,G^{(k)}) \left(\frac{1}{V}\right)^{N-|\mathcal{F}|}} dx^t \\
&= \int x^t_M f_{\mathbf{x}^t_{\mathcal{M}}|\mathbf{x}^t_{\mathcal{F}}\boldsymbol{\theta}^t\boldsymbol{\phi}^t\mathbf{G}}(x^t_{\mathcal{M}}|y^t_{s_{\mathcal{F}}},\theta^t,\phi^t,G^{(k)}) dx^t_{\mathcal{M}} \\
&= \mu^{(k)}_{\mathbf{x}_{\mathcal{M}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t} + \Sigma^{(k)}_{\mathbf{x}_{\mathcal{M}}\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t} \left[\Sigma^{(k)}_{\mathbf{x}_{\mathcal{F}}\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t}\right]^{-1} \left(\mu^{(k)}_{\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t} - y^t_{s_{\mathcal{F}}}\right).
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{M}} {x^t_{\mathcal{M}}}^T\right] &= \Sigma^{(k)}_{\mathbf{x}_{\mathcal{M}}\mathbf{x}_{\mathcal{M}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t} - \Sigma^{(k)}_{\mathbf{x}_{\mathcal{M}}\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t} \left[\Sigma^{(k)}_{\mathbf{x}_{\mathcal{F}}\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t}\right]^{-1} \Sigma^{(k)}_{\mathbf{x}_{\mathcal{F}}\mathbf{x}_{\mathcal{M}}|\boldsymbol{\theta}^t\boldsymbol{\phi}^t} \\
&\quad \ldots + \mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{M}}\right] \mathbb{E}_{q^{(k+1)}}\left[{x^t_{\mathcal{M}}}^T\right].
\end{aligned}$$

Now, without loss of generality, we can write

$$x = \begin{bmatrix} x^t_{\mathcal{F}} \\ x^t_{\mathcal{M}} \end{bmatrix}$$

and obtain

$$\begin{aligned}
\mathbb{E}_{q^{(k+1)}}\left[x^t\right] &= \begin{bmatrix} y^t_{s_F} \\ \mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{M}}\right] \end{bmatrix} \\
\mathbb{E}_{q^{(k+1)}}\left[x^t x^{tT}\right] &= \begin{bmatrix} y^t_{s_F} y^{t\,T}_{s_{\mathcal{F}}} & y^t_{s_{\mathcal{F}}} \mathbb{E}_{q^{(k+1)}}\left[x^{t\,T}_{\mathcal{M}}\right] \\ \mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{M}}\right] y^{t\,T}_{s_F} & \mathbb{E}_{q^{(k+1)}}\left[x^t_{\mathcal{M}} x^{t\,T}_{\mathcal{M}}\right] \end{bmatrix}
\end{aligned}$$

which completes our derivation.

Chapter 4

Experiments

4.1 Detection and Labeling with EM

4.1.1 Loopy Graphical Model and Global Variables

In Sections 3.2.4 and 3.3.4 we have presented our extensions to the triangulated decomposable model of [SGP00]. These can be divided into two main areas: the structure of the graphical model, and the invariance with respect to translations. In the following experiments we want to understand the benefit (if any) of our modeling efforts.

The system we analyze takes as input a set of features (and their instantaneous velocities) that have been detected on a video frame. Since we wish to have a quantitative measure for the labeling performance, we use a dataset for which a ground truth labeling is available. This is obtained by means of a motion-capture system, which records a collection of point-features attached to the joints of an actor. This setup allows us to cause the disappearance of some of the body parts in a frame. We use this to simulate occlusions for which we test the robustness of the system. Another available manipulation is the introduction of spurious detections, that is, features that do not belong to the body. This allows us to simulate real-world conditions for a feature detector (such as the of [TK91], for example), where a large number of features detected are of no interest to the system.

The system is meant to perform two tasks, which are interconnected. The first one is the *detection* task: given a set of N features, is there a human present? The second one, which is only of interest when the first task has an affirmative answer, involves finding the subset of the N detections which most likely matches the structure of the human body.

In our experiment we use two sequences W1 and W2 of 7,000 frames each[1]. Both sequences depict a person, seen from the side, walking back and forth on a line. For each pair of frames in the sequence, we use the labels obtained with the motion capture system, and compute the instantaneous velocity of each body part. This results in a simulated Johannson's display with positions and velocities of a set of features.

4.1.2 Dealing with Occlusions

We use the sequence W1 to learn the model's parameter and structure, according to the procedure described in Section 3.4.1. Since our model is probabilistic, the task of detection is accomplished by returning a probability of the human being present. To test the robustness of the system with respect to occlusions, we vary the number of visible points to the algorithm. This is done for both the hand-made triangulated decomposable graph with local translation invariance, presented in [SGP00], and our EM-based inference scheme on a loopy graph, with the centroid providing global translation invariance. The ROC plots in Figure 4.1 were obtained by first presenting 30 points of clutter, together with 4, 7, or 11 body parts (out of 14 available), chosen at random in each frame. We then run the algorithm with only 30 points of clutter. In each frame the algorithm performs a labeling and detection task, returning a number P_o in $[0, 1]$ which represents the confidence that a human being is present. By setting a threshold $T \in [0, 1]$ on P_o we can interpret the answer

[1]We thank the authors of [SGP00] for making the data available to us.

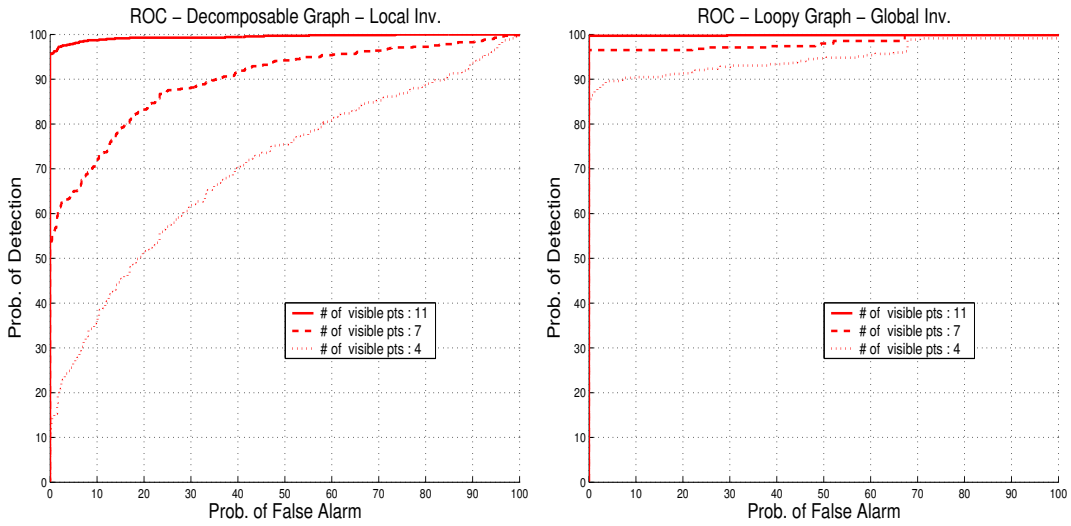


Figure 4.1: ROC Curves. [Left] Performance of the hand-made decomposable triangulated graphical model with local translation invariance of [SGP00]. [Right] Performance of our loopy graphical model with global translation invariance. All the curves are obtained by changing (over the range $[0, 1]$) the threshold on the probability returned by the algorithm. For each value of the threshold we plot the fraction of times the algorithm correctly claims to have detected a person when the display shows one ($P_{detection}$), versus that of mistakenly stating that a person is there when only 30 points of clutter are presented to the algorithm ($P_{false-alarm}$). Varying the number of visible points between 4, 7, and 11 gives the dotted, dashed, and solid lines respectively.

of the algorithm as a binary choice (either presence or absence of the human is claimed) in each frame. Over all the frame, a certain fraction will be correctly identified as containing a human (since one was there), while some will be tagged as containing one even though only clutter was shown. These two numbers are called the *probability of detection* ($P_{detection}$) and *probability of false alarm* ($P_{false-alarm}$). Varying the threshold from 0 to 1 changes both $P_{detection}$ and $P_{false-alarm}$, producing one of the curves of Figure 4.1. By observing the ROC plots, we notice that our model achieves a significant improvement in robustness with respect to occlusions. This is obtained by representing more general connectivity, and relying on the global centroid for translation invariance.

4.1.3 Labeling Accuracy

Next, we investigate the accuracy of the labeling procedure. As in the previous experiment, we use the sequence W1 to learn the model's parameter and structure. A 700-frame random sample from the sequence W2 is then used for testing of the algorithm.

We evaluate the performance of the EM-based inference on loopy graph with the global centroid, and compare it with the hand-made, decomposable graphical model of [SGP00]. For each one of the testing frames, we let the algorithm label the body parts. Over all the frames, we compute the percentage of labels that were incorrectly assigned. A curve in Figure 4.2 is obtained by varying the number of parts that are presented to the algorithm. An additional 30 points of clutter is part of the input in every frame.

We first explore the benefits of just relaxing the decomposability constraint, still implementing the translation locally. The lower two, dashed curves of Figure 4.2 already show a noticeable improvement, especially when fewer body parts are visible. However, the biggest achievement is brought by global translation invariance, as is evident from the upper two curves of Figure 4.2.

4.1.4 Computational Issues

Although the theoretical computational complexity of our loopy belief propagation (LBP) inference is of the same order of magnitude as that of the dynamic programming (DP) algorithm of [SGP00], a few observations should be made. Firstly, the DP procedure is exact and guaranteed to complete inference in one iteration of message passing. This is not the case for LBP, which takes 4 to 5 iterations to converge.

Secondly, to avoid local maxima during the EM procedure, we restart the inference algorithm at most 10 times, using a randomly generated schedule to pass the messages.

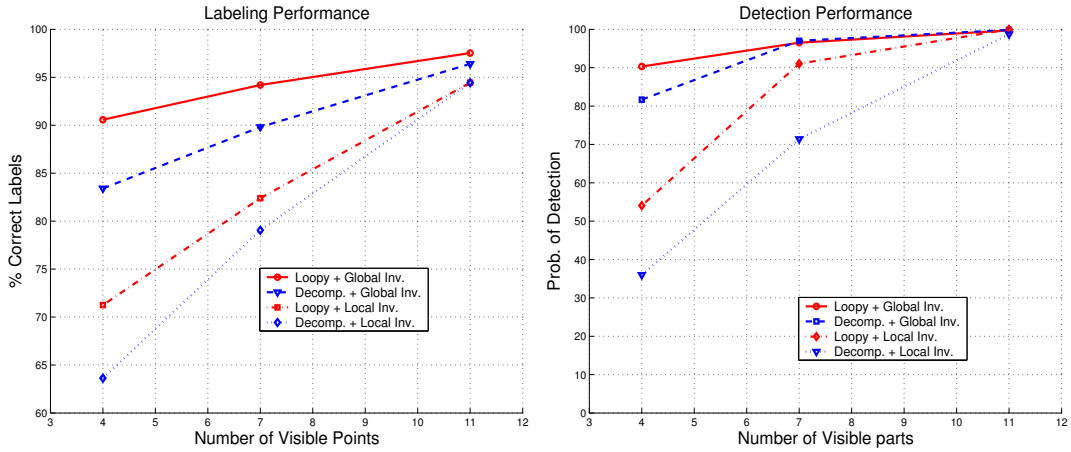


Figure 4.2: Detection and Labeling Performance. [Left] Labeling: On each display from the sequence W2, we randomly occlude between 3 and 10 parts and superimpose 30 randomly positioned clutter points. For any given number of visible parts, the four curves represent the percentage of correctly labeled parts out of the total labels in all 700 displays of W2. Each curve reflects a combination of either local or global translation invariance and decomposable or loopy graph. [Right] Detection: For the same four combinations we plot $P_{detection}$ (probability of detecting a person when the display shows one) for a fixed $P_{false-alarm} = 10\%$ (probability of stating that a person is present when only 30 points of clutter are presented). Again, we vary the number of visible points between 4, 7, and 11.

Finally, when global invariance is used, we reinitialize the value of the centroid $\boldsymbol{\theta}$ up to 10 times. Each time we randomly pick a value within a different region of the display.

On average, about 5 restarts for $\boldsymbol{\theta}$, 5 different schedulings, and 3 iterations of EM suffice to achieve a labeling with a likelihood comparable to the one produced by ground truth labeling.

4.2 Detection and Labeling with Sample Propagation

In this set of experiments we analyze the performance of our system in a more challenging setting. The dataset we use is comprised of several real-world-conditions video sequences recorded with a standard camcorder (see Figure 4.3 to see an example of our dataset). We also replace the EM-based inference algorithm of Section 3.3.4.1 with the more efficient

Figure 4.3: Data. (a) An example video frame from the training video sequence. (b)–(f) Example frames from the testing data, including various types of motions performed by different objects/people with various appearances (clothing).

and faster sample propagation, which we presented in Section 3.3.4.2. Finally, we test our algorithm recognition capability by using more challenging and confusing clutter inputs.

In order to provide our algorithm with a Johannson-like input display, we process our video sequences with a Lucas-Tomasi-Kanade (LTK) feature detector and tracker [TK91]. The set of feature points (locations and velocities) thus obtained is the input to our algorithm.

4.2.1 Speeding-Up the Learning with Appearance

Since the clothing and the environment remain the same throughout a sequence, we incorporate appearance information in the description of each part, and model it as an isotropic Gaussian, which we multiply into the geometry portion of the density for that part.

The model structure and parameters are learned from data without supervision. We use a training set made up of 378 frames, taken from a single video sequence, showing a

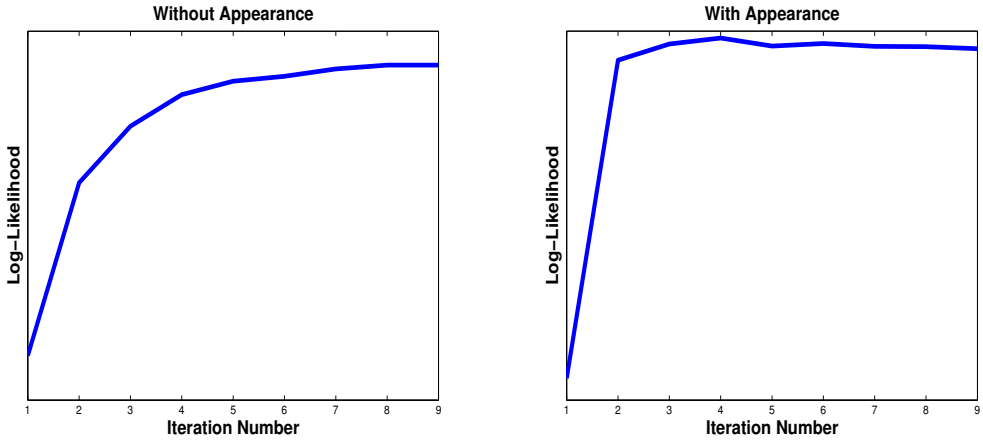


Figure 4.4: Learning Speed. A comparison of the log-likelihoods while learning a model with and without appearance. Using appearance information the model converges significantly faster. The log-likelihood is not monotonic since, due to efficiency, we use an approximated algorithm to compute it.

single person walking from right to left, parallel to the camera image plane. All the frames were taken with the camera at the same view point and with the person wearing the same set of clothes.

In training we ignore all points on the background (using background subtraction). We extract an 11×11 patch around each detection point and use a 3D histogram with 64 bins to represent the distribution of colors (in HSV space) of the patch's pixels. The vectorized histogram is used as the appearance representation of the detected point.

In this first experiment, we learn the hybrid probabilistic model with $M = 12$ parts using the approach presented in Section 3.4.1. Figure 4.4 compares the log-likelihood of the learned model while using (or not) appearance information. To learn a model using geometric information alone, we simply drop the appearance factor from the probability density function of each part.

It can be seen that using appearance results in significantly higher convergence rate, that is, the appearance of the detections reduces the probability of wrongly assigning body

parts to detections. Note that the actual values of the log-likelihood in the two cases are not comparable (as one likelihood is estimated while including appearance and the other one without it). We thus placed them on two separate plots and removed the log-likelihood values to avoid confusion.

4.2.2 Recognition Accuracy

Next, we examine the ability of our model to detect a specific action among others. Given a video sequence, the task is to recognize the presence of a human performing a right-to-left walk.

As before, the model structure and parameters are learned from data without supervision. The training set made of 378 frames is taken from a single video sequence, and shows a single person walking from right to left, parallel to the camera's image plane.

The testing set included 2688 frames taken from 14 sequences of various lengths. A total of 1101 frames were representative of right-to-left walking, while 1587 frames depicted other types of motion, including running, cycling, a car driving by, water moving, and walking left to right. Since our system is not scale invariant, during our preprocessing step we scaled the sequences, so that the height of a person would be similar in all of them.

We compare three modes of learning and recognition:

- Learning and recognition without appearance [blue]
- Learning with appearance, and recognition without appearance [red]
- Learning and recognition with appearance [black].

In Figure 4.5 we report the three ROC curves obtained by varying the decision threshold from 0 to 1. It can be seen that using appearance in both learning and testing resulted

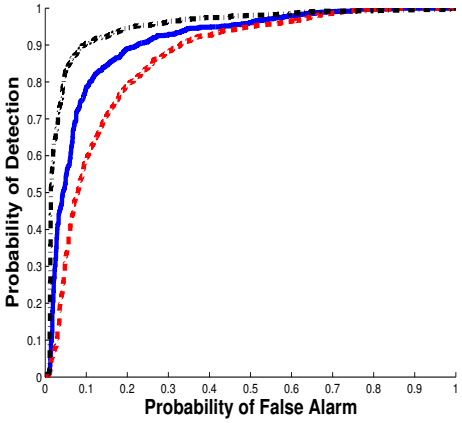
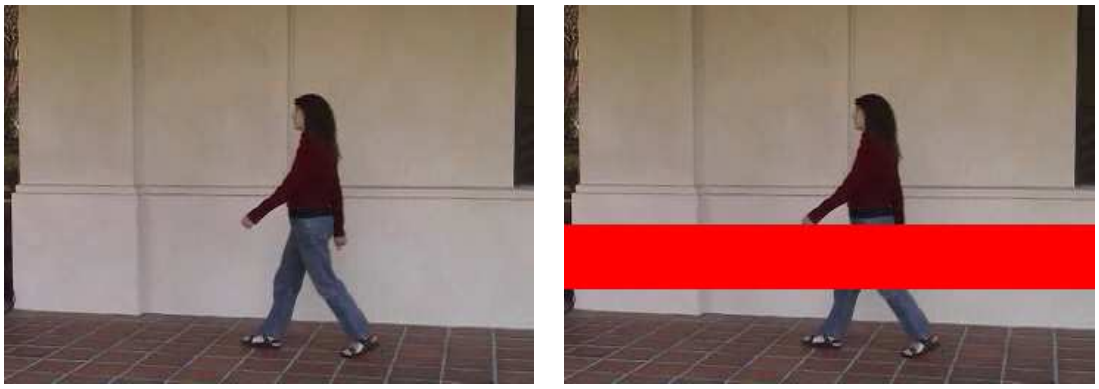


Figure 4.5: Recognition Results. A comparison of ROC curves corresponding to the three modes of experiments with reference to appearance. In blue is learning and recognition without appearance. In red, we show learning with appearance, and recognition without appearance. Finally, in black, we use appearance in both learning and recognition. This last one yields the best recognition results.

in significantly higher performance. Interestingly, the poorest results were obtained when learning with appearance and testing without appearance. We believe the reason for this is that the model was optimized for the use of both geometry and appearance, hence having to (in part) trade off fitting the geometry information alone, in favor of the appearance. Yet, during testing the geometry part of the model was the only one available.

4.2.3 Robustness to Occlusions

To conclude we show how, with the use of the global centroid and appearance, the model achieves robustness to occlusions. To test this aspect we selected three video sequences, with a total of 919 frames, and added to all their frames a virtual occlusion which hides the thighs, as shown in Figure 4.6 (b). We then compare the recognition rates obtained for the same sequences with and without the occlusion, using appearance in both learning and recognition. To compute recognition rates we need to select a threshold on the likelihood output by the system, and build a binary classifier to tag all frames. We set the threshold



(a) (b)

	Recognition Rate
Without Occlusion	94.02%
With Occlusion	91.3%

(c)

Figure 4.6: Comparison of Recognition Rates. We provide the input data to the algorithm with and without occlusions. In (a) we show an example frame from the test dataset. In (b) the same frame is shown after introducing an occlusion over the thighs. The table (c) summarizes the recognition rates.

according to the ROC curve of the model which uses appearance in both learning and recognition. The threshold is taken as the "equal error" detection rate, that is, the value of the threshold for which $P_{detection} = 1 - P_{false-alarm}$. The results, summarized in Figure 4.6 (c), show that only a slight decrease in performance occurred in recognition on the partially occluded frames.

Chapter 5

Movemes, Actions, and Activities

5.1 Introduction

One of the goals of the computer vision community is to endow computers with the ability to observe and *understand* the motion of humans. If we could build new machines that autonomously interact with us in our environment, without the need for us to actively declare our intentions and behaviors, a large number of applications would immediately be possible.

Understanding what a human is doing, even in a confined setting, is a challenging task for a number of reasons. Motion happens (and its semantic is encoded) at different scales, both in space and time. The body is a highly deformable object, and can present itself in an extreme variety of poses and appearances. Even more challenges arise when we deal with a moving camera, a non-stationary background or occlusions of the body by other objects or the body itself. If that was not enough, the underlying intentions might be ambiguous, and their interpretation could require abstract knowledge not readily available from the video signal [ZT01].

Notwithstanding these issues, the problem of understanding human behavior has received a lot of attention in the last thirty years. In the following sections we present our

view on the problem, and we introduce two approaches to the construction of a simple dictionary of elementary motions, which we see as the initial step toward the comprehension of human behavior. For a broad review of the existing literature in human motion, we refer to a few survey papers such as [AC99, BD01, Bob97, Gav99].

5.2 A Hierarchical View

One major obstacle that even we, as humans, encounter when observing people's behavior, is given by the difficulty of verbalizing and communicating in detail what we see. If we limit ourselves to everyday *activities*, we are generally able to provide a coarse description of what happened in a given period of time. This is usually achieved by listing verbs which represent the *actions* of interest. The problem becomes slightly more complex when we are asked to describe how specific *actions* are done. For example, if we had to convey what is involved in cooking a particular dish, we could present "mixing the ingredients" as a repetitive task where the arm makes a circular motion, while holding a wooden spoon, etc. Yet, other actions—such as jumping to hit the ball during a volleyball game—might be more challenging, since a precise combination of precisely executed *atomic motions* is the only way to achieve the desired goal.

If we move to an even finer scale, describing the actual motion of our limbs and body parts to others with the purpose of having them repeat the same or similar trajectory, we are left with little choice but to show an actual example of what we mean. Aside for a few motions for which we have words like "step", "reach", "grab", "swing", etc., we suffer from the lack of a taxonomy that appropriately represents what we understand and interpret effortlessly.

Although some attempts have been made (see, e.g., Rudolf von Laban's 1928 work) to

develop a notation that describes human motion, the direct mapping to visually observable aspects is still missing. The general consensus of the community seems to be that a fixed vocabulary for human motion is both non-existent and inappropriate. This is also our opinion, and we essentially agree with [Bob97] that the right approach must be hierarchical, with different taxonomies at different levels of abstraction. We base our decomposition of human motion primarily on the time-scale at which it happens, and in part on the semantic we commonly attach to it.

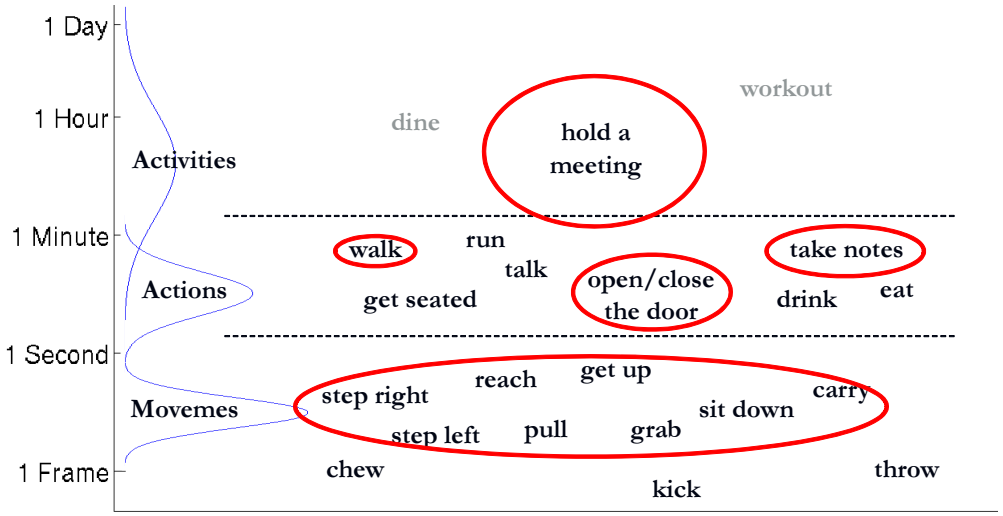


Figure 5.1: A Hierarchical View. We interpret human motion in a hierarchical way. At the highest layer, a single word is sufficient to provide a compact description of an "activity". We use the term "action" for shorter events that, joined together probabilistically, yield an activity. At the bottom layer are the "movemes": atomic motions which are learned without supervision from data, and do not necessarily posses a verbal description. We arbitrarily name them for the sake of example.

At the highest level, we think of *activities* as happening over an extended period of time. The top portion of Figure 5.1 shows a few examples of what could be regarded as an activity. Humans generally have no trouble choosing words that identify an activity, and the choice is generally agreed upon by the majority, leaving no ambiguity.

The middle layer represents brief events, which we call *actions*, that usually last up to

a few seconds. Repetitive events such as walking are also classified as actions, since the elementary cycle that is repeated fits the definition of action. Another defining aspect of actions is that their combination, meant in a stochastic manner, yields an activity. For example, the activity of "holding a meeting" could be considered a probabilistic concatenation of actions such as "sitting down", "opening a door", "taking notes", etc. In general, the duration, the order (to some extent), and even the presence or absence of some of the actions do not compromise the existence of the activity.

Elementary pieces of motion, limited in time to a few frames, which could even involve only part of the body (such as a limb, or the lower/upper body), belong to the bottom level. Inspired by the early work of Bregler [Bre97], we refer to them as *movemes*. Although we are able to associate names for some of these elementary motions, we do not feel this needs to be the case. In fact, unlike actions and activities, we believe that their definition should be inferred from data as the "best" set of short-term motions describing the actions observed. Unfortunately the definition of "best" here is vague, since it is difficult to develop a meaningful metric that makes sense for every problem. Nevertheless, guided by a few requirements that we deem important, we propose a procedure for the identification of a dictionary of movemes in video sequences.

5.3 A Simple Approach to Discovering Movemes

In Chapter 3 we have seen how the expectation-maximization (EM) algorithm is a versatile procedure that can be applied to a number of problems. By taking advantage of the decoupling induced by latent variables, it is possible to alternate between two simpler optimizations, leading to a solution for the original problem. In the following section we present our first attempt at building a dictionary of movemes, which is based on the EM algorithm.

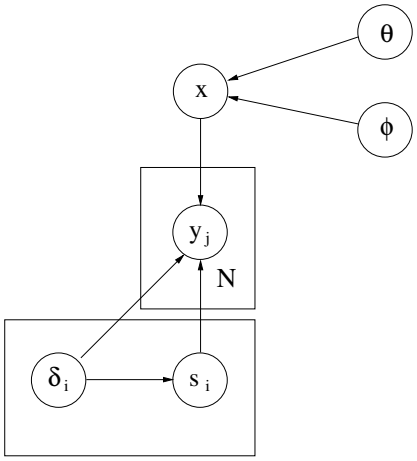


Figure 5.2: Graphical Model. We model a collection of parts $\mathbf{x}$ which are put in correspondence with a subset of the detections $\mathbf{y}$ by the labeling variable $\mathbf{s}$ and $\boldsymbol{\delta}$. The global variables $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ represent the centroid of the body and the index of the moveme.

5.3.1 Learning Movemes with Expectation Maximization

The setup we use is similar to that of Section 3.2.1. A collection of features has been automatically detected on each frame, and their description represented by a small vector of parameters (e.g., position, velocity, average color of surrounding patch, etc.).

For convenience, we report in Figure 5.2 the graphical model which we have abundantly discussed. Recall that $\mathbf{x}$ models the body parts, which are put in correspondence with a subset of the detections $\mathbf{y}$ by the labeling variable $\mathbf{s}$ and $\boldsymbol{\delta}$. The global variable $\boldsymbol{\theta}$ represents the centroid of the body, while $\boldsymbol{\phi}$ is the index of the moveme.

After marginalizing with respect to the parts $\mathbf{x}$ we can write the following factorization

$$f_{\mathbf{yh}\boldsymbol{\theta}\boldsymbol{\phi}}(y,h,\theta,\phi) = f_{\mathbf{x}_{\mathcal{F}}|\boldsymbol{\theta}\boldsymbol{\phi}}(y_{s^f}|\theta\phi)\left(\frac{1}{V}\right)^{M-|\mathcal{F}|} f_{\boldsymbol{\theta}}(\theta) f_{\boldsymbol{\phi}}(\phi) f_{\mathbf{h}}(h) \tag{5.1}$$

where, once again, we use $\mathbf{h}$ in place of the labeling variables $\mathbf{s}$ and $\boldsymbol{\delta}$.

In the detection and labeling problem of Chapter 3 we wanted to recognize a specific action (e.g., walking from right to left) among other distractors. Here instead, we focus on

the problem of learning movemes, without supervision, from a corpus of videos. To achieve this, we use the procedure of Section 3.4.2, where the input to the algorithm is a dataset with multiple actions being performed. The role of the global variable $\boldsymbol{\phi}$ is now easier to understand: we specify the number of movemes to be learned, and the model tries to fit the data by segmenting it in time and probabilistically assigning frames to each of the "classes" indexed by $\boldsymbol{\phi}$. The unsupervised learning and inference procedures of Chapter 3 are readily applicable, since they have already been derived in their most general form, which takes into account the global variable $\boldsymbol{\phi}$.

We observe that the estimate of $\boldsymbol{\phi}$ is performed on a frame-by-frame basis. As consequence, there is in principle no guarantee of continuity for the value of $\boldsymbol{\phi}$ over time. On the other hand, we would like to enforce the fact that movemes (as we defined them) have a typical duration which ranges from a few frames to about a second. To achieve a segmentation which is consistent with this definition, we smooth the resulting estimate of $f^{(k)}_{\boldsymbol{\phi}|\mathbf{y}}(\phi|y)$, after each iteration of EM, using a triangular kernel with a limited support (typically eight or ten frames) around its center.

5.3.2 Datasets and Features

In our first attempt to build a dictionary of movemes, we use several video sequences of aerobic exercises. The idea of discovering movemes without supervision relies on the fact that data is available which depicts a sufficiently large variety of actions. Since fitness programs are designed with the goal of moving as much of the body as is possible, we feel they are a good choice for applying our procedure. Additionally, the exercises are often repeated several times, allowing for multiple instances of the same moveme/action to be presented to the learning algorithm.

Figure 5.3: Manually Labeled Data. We show three sample frames from our input data. The marked boxes were automatically identified by the box detector of Section 2.2, and manually selected, out of all the available ones. The frames are cropped for better visualization.

Since in this part of our work, we have shifted our focus on the temporal segmentation of the sequences, rather than their labeling, we choose to avoid the task of unsupervised labeling of the data, and rely on the matching of body parts and detections provided by an operator.

To make this process easier, and minimize the amount of human labor, we pre-process the videos with a feature detector which identifies approximately rectangular regions on the image. This is done according to the procedure described in section 2.2.

Since the background is stationary in our video sequences, we can establish an approximate bounding box around the moving person. We then limit the amount of detections per frame by retaining only those which appear in the neighborhood of the person. Finally, the operator identifies and labels the detection that best matches the underlying body parts. This results in incomplete data, from time to time, since the body part may be occluded, or the detector can fail to produce a suitable candidate. The EM procedure, however, can cope with this and we are able to learn a model from a dataset thus obtained. Figure 5.3

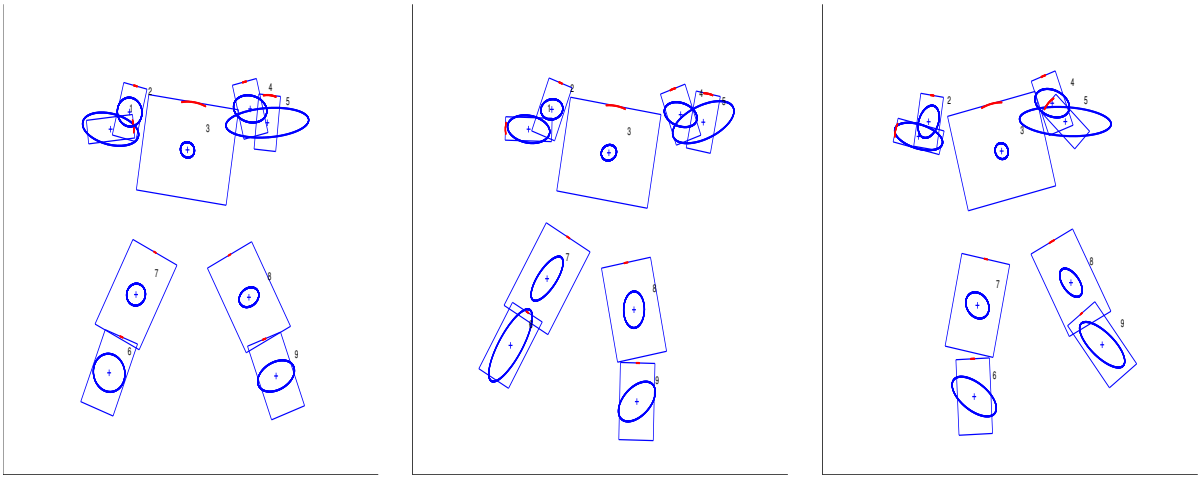


Figure 5.4: (E1) Automatically Learned Models. A 3-movemes model, learned from 2 video sequences totaling 469 frames, whose components correspond to the three principal motions that the sequence displays. The ellipses mark the position and covariance of each body part relative to the centroid. The boxes correspond to the mean width, height, and angle of the body parts, i.e., the mean pose of the body parts during these motions.

gives an example of typical frames, after they have been processed for learning.

It is important to note how, in the detection and labeling problem, the motion information (in the form of location and instantaneous velocity) was encoded in the descriptor for each feature detected, while the identity of the features was unknown. Here instead, we are looking at a different problem: the identity of the features is known, while we wish to discover the classes of motion the body undergoes.

5.3.3 Experiments

In the following experiments we present the results of our algorithm on a few datasets.

(E1) Learning Movemes—Dataset 1 :

In this experiment we selected 2 video sequences, totaling 469 frames, showing three different motions. Since the motions do not have a "conventional" name, we call them lift-left-leg, lift-right-leg, and rotate-upper-body.

As noted before, we used the box detector to find candidate body parts in all the frames. An operator hand labeled the body parts in each frame. We then ran our learning algorithm allowing it to fit 3 phases to the data. Although we have experimented with $n_\phi = 5$ and 7 phases, we found that the aforementioned three movemes gave the most stable and interesting representation. Figure 5.4 shows the resulting model. The ellipses mark the position and the one standard deviation uncertainty of each body part relative to the centroid. The boxes correspond to the mean width, height and angle of the body parts during the motion. The red arc at the top of each box represents the angle uncertainty. To avoid cluttering the figure, no uncertainty on width and height is reported.

(E2) Testing Movemes—Dataset 1 :

Following the first experiment, we took the model obtained and we evaluated the analysis power of this approach on a total of three sequence, including the 469 training frames plus an additional 500 unseen frames.

This time we allowed all the boxes detected in every frame (not just the few which we hand labeled) to be presented as input to the algorithm. The task is to find, for each frame, the configuration of boxes that is most probable in the model, as well as the temporal segmentation into movemes. Figure 5.5 displays a few representative frames with all the detected box overlaid (marked in blue). The boxes marked in red are those selected by the algorithm as the best assignment to body parts. Having used real-world conditions data, we do not have a complete ground truth to perform a qualitative evaluation of the labeling performance. For the segmentation results, at least at the level of movemes, quantitative analyses are of little value, since the

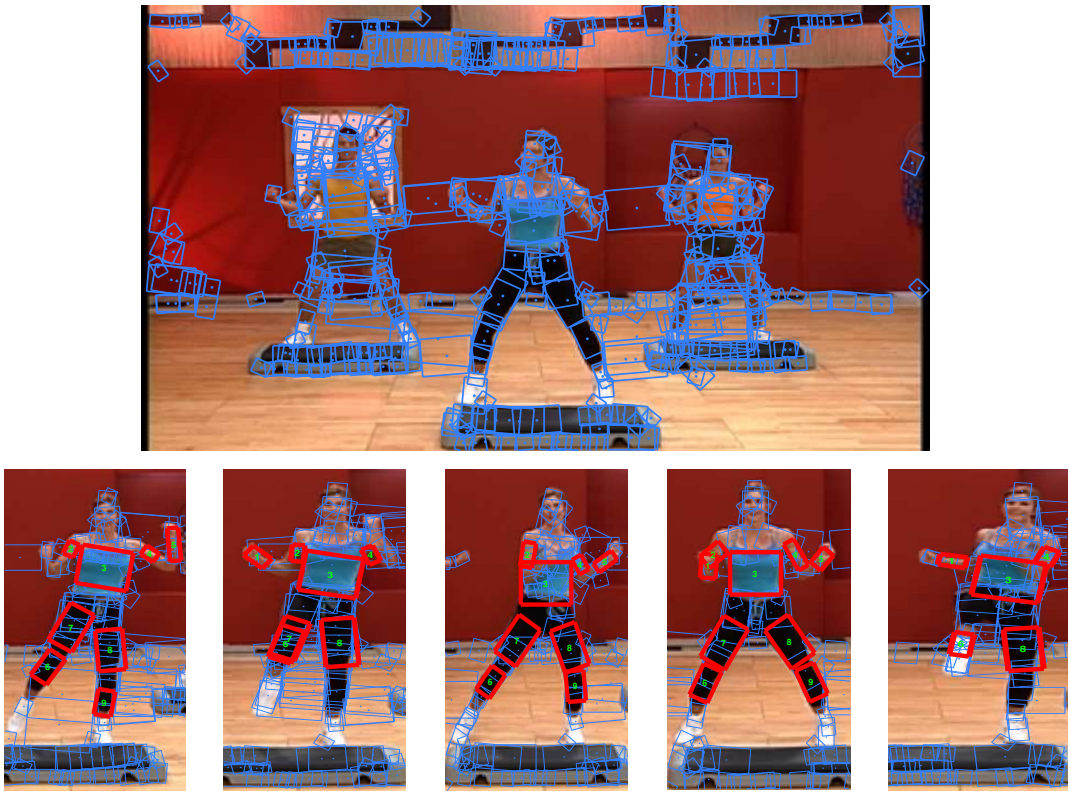


Figure 5.5: (E2) Testing the Movemes. Sample frames from the 968-frame testing sequences. Provided as input were all the detected boxes in each frame, which are marked by thin blue lines. The top figure shows a full frame. At the bottom we show five sample frames. A thick red line marks the boxes that were chosen by the algorithm as representing the best human-like configuration. The frames have been cropped to show only the labeling results.

decision as to when one motion ends and another starts is itself a matter of debate among humans.

Figure 5.6 displays temporal segmentation into movemes. On the horizontal axis we report the time, or frame index. For each frame t, on the vertical axis we represent the probability of each of the three phases. The thicker the bar corresponding to one of the three phases, the more likely the frame is to belong to that phase, with the total thickness being fixed (and corresponding to probability 1). The color segmentation along the bottom axis represent *a* ground truth, which is provided by a human. The same colors have been assigned (a posteriori) to the three movemes/phases identified

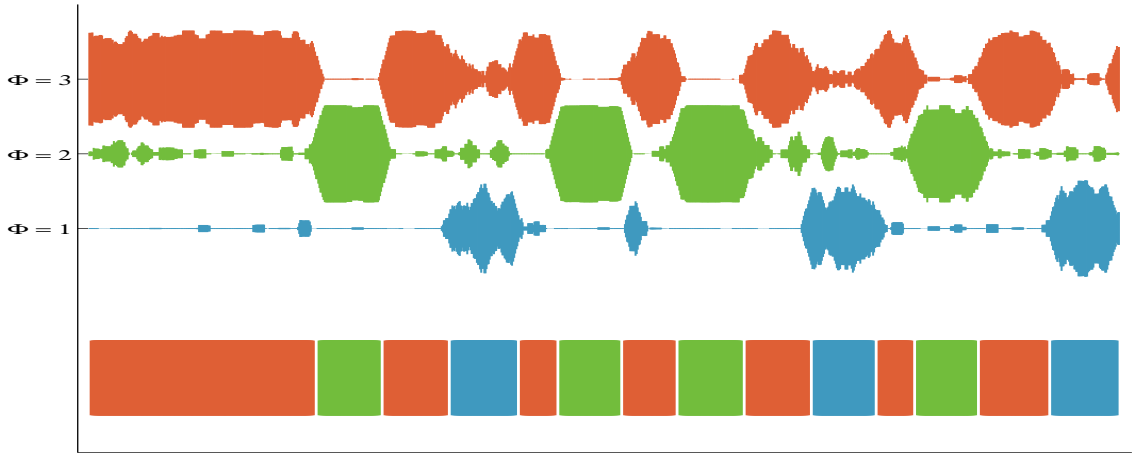


Figure 5.6: (E2) Likelihood of Movemes. On the horizontal axis we report the time, or frame index. For each frame, on the vertical axis we represent the probability of that frame to be assigned to each of the three phases. The thicker the bar corresponding to one of the three phases, the more likely the frame is to belong to that phase. The color segmentation along the bottom axis represent "*a* ground truth", which is provided by a human who was unaware of the result of the experiment. The same colors have been assigned (a posteriori) to the three movemes/phases identified by the algorithm, by observing the best match between the movemes and the human-generated segmentation.

by the algorithm, by observing the best match between the movemes and the human-generated segmentation.

(E3) Learning Movemes—Dataset 2 :

A second learning experiment was performed on a different set of sequences with a total length of 1129 frames. These videos show a more complex set of aerobic exercises. Each exercise is different, yet some of the movements are similar among exercises. For example one exercise is composed of raising the arms up, crossing the arms in front of the chest, and letting them down. Another one involves stretching the arms out, crossing them in front of the chest again, and finally pulling the hand towards the shoulder. Clearly, the parts where the arms are crossing in front of the chest is shared between the two exercises. This is nicely captured by the 7-component model that the algorithm learns.

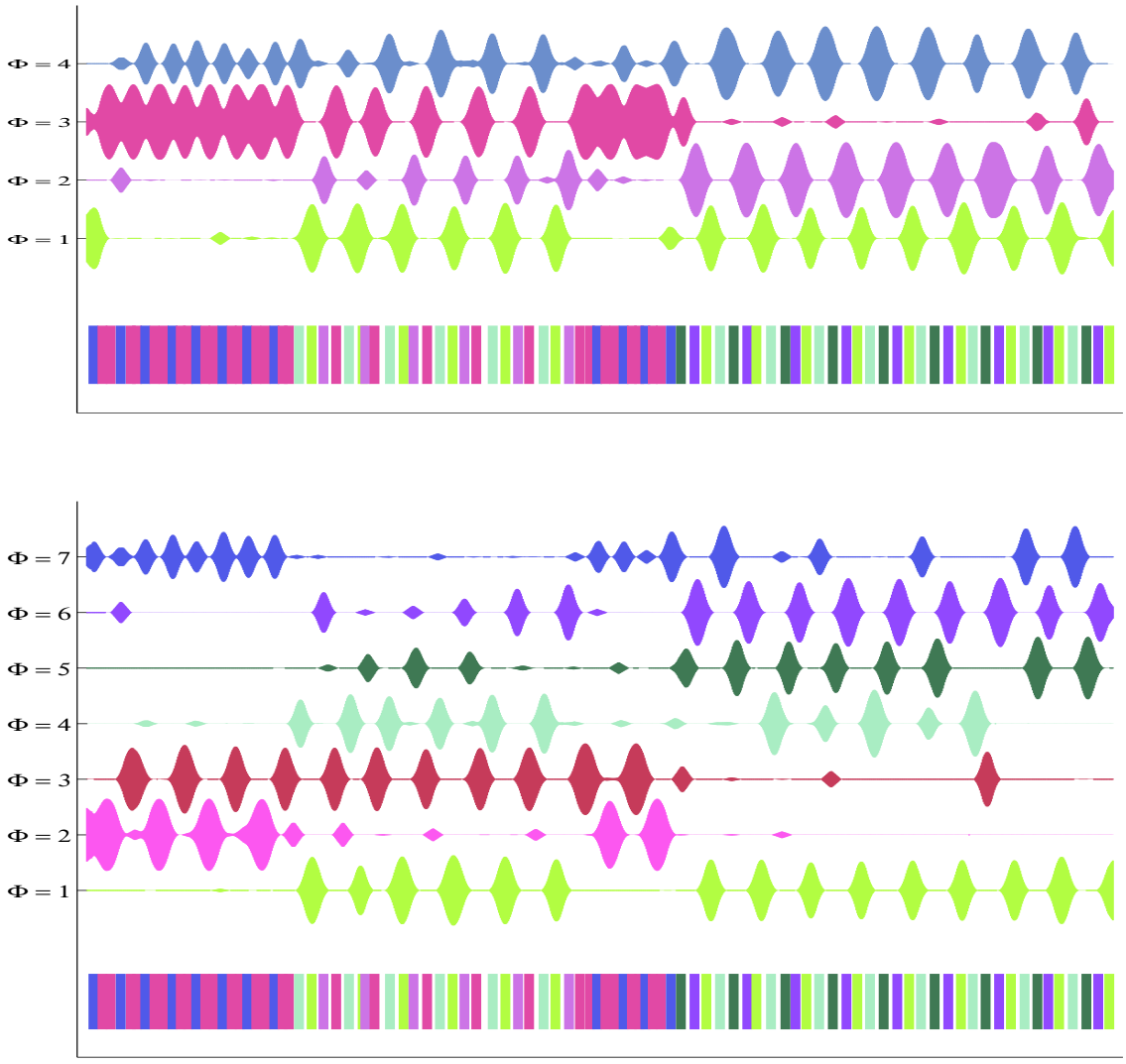


Figure 5.7: (E3) Choosing the Number of Components. Motion analysis results for the second dataset with 4 and 7 components. Colored bars on the bottom row indicate human-made segmentation at the turning points between aerobic moves. The colors used for each moveme have been chosen (a posteriori) so as to enhance the matching between the ground truth and the algorithm's choice of movemes. Major changes in the type of exercise can be qualitatively detected in either plot by simply looking at the change in temporal patterns. A more careful analysis of the above diagrams, in conjunction with the video, shows (as expected) how different complex actions share some of the same phases. These correspond to shared movemes such as crossing of the arms. When using fewer phases (top) not all the movemes in the sequence are captured and different actions appear as similar ones. Conversely, when using a higher number of phases the segmentation exhibits multiple phases trying to "explain" what a human perceives as the same motion.

One of the unsolved issues of our approach is the choice of the number of components.

To explore the influence of this parameter we have learned, for these sequences, models

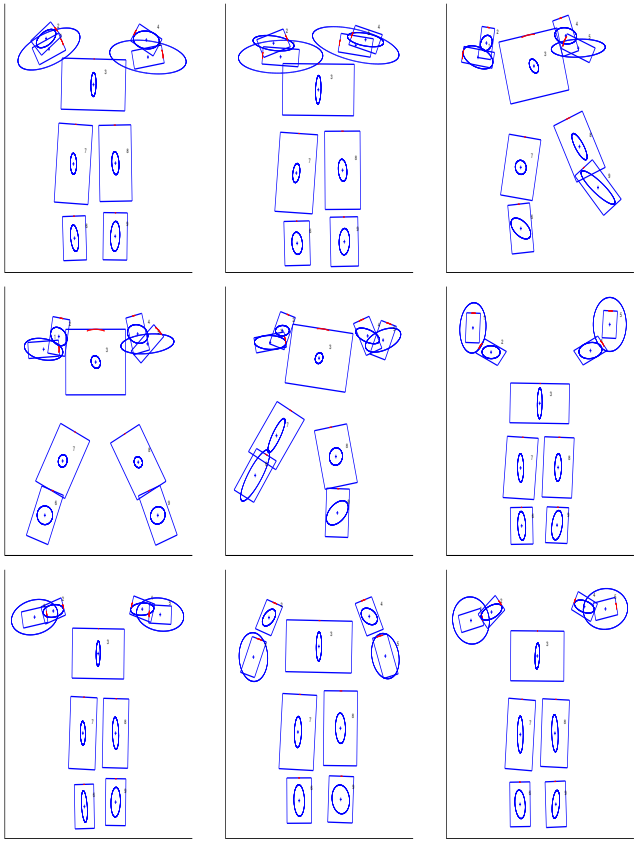


Figure 5.8: (E4) Automatically Learned Models. The 9-moveme model learned from a few sequences totaling 1629 frames depicting different actions. The ellipses mark the position and covariance of each body part relative to the centroid. The boxes correspond to the mean width, height, and angle of the body parts, i.e., the mean pose of the body parts during the motion. The arc at the top of each box represents the angle covariance. To avoid cluttering the figure, no uncertainty on width and height is reported.

with n_{ϕ} between 3 and 9. Figure 5.7 displays the probability density of the phase over time for two choices of n_{ϕ} (namely, $n_{\phi} = 4$ and $n_{\phi} = 7$). Understandably, the larger the number of components, the finer the decomposition into movemes that we obtain. On the other extreme, when the number of components is too small (like in the case $n_{\phi} = 4$), movemes that a human tend to classify as different, are merged together and represented by a single component.

(E4) Learning Movemes—Dataset 1 and 2 :

In our largest learning experiment we trained a 9-component model on all of the

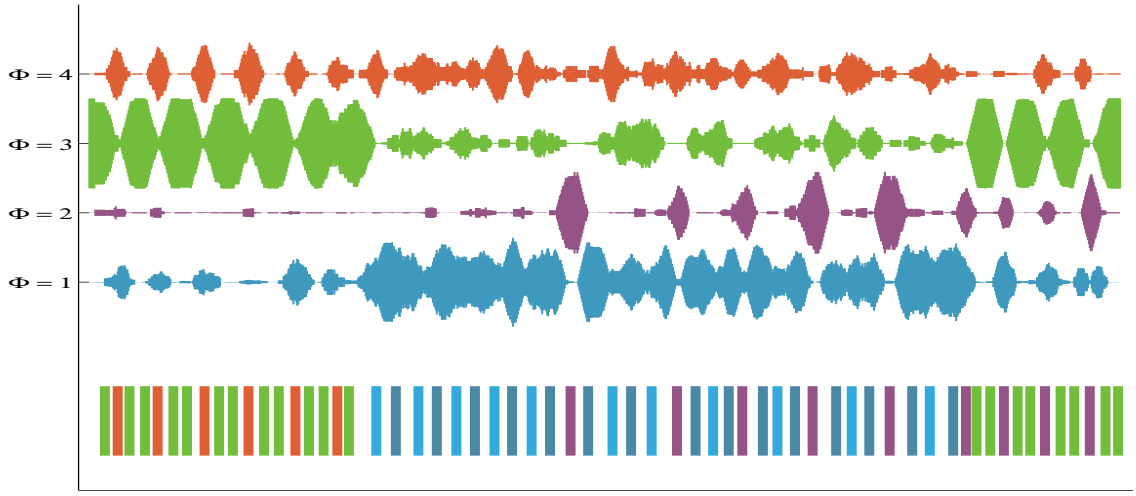


Figure 5.9: (E5) Testing Movemes. An actor performs a number of exercises which are being analyzed by a 4-component model. The training was done on a sequence of similar exercises performed by a different actor. Some of the phases follow a repetitive pattern, showing that the action performed is based on movemes that were present in the training set. More uncertainty in other phases is due to the ambiguities in the way the action is performed (e.g., arms are neither straight down nor wide open, so two or more movemes are trying to explain that configuration).

1598 labeled frames. The schematic graphical representations of the components (shown in Figure 5.8) exhibit a good qualitative correspondence with the actions in the sequences.

(E5) Testing Movemes—Dataset 3 :

In our final experiment, we evaluated the capability of this approach to describe actions performed by a different actor in an unlabeled sequence of 1057 frames. The exercises presented are similar (but not identical) to the ones used for training, which was done as in (E3). Figure 5.9 shows the resulting probability for each moveme on the testing sequence. Some of the phases show a high probability, and follow a repetitive pattern, indicating that the action performed is based on movemes which are being recognized as similar to those present in the training set. The higher uncertainty over the phases, which we find in the central part of the sequence, is due to the ambiguities

in the way the action is performed (e.g., after crossing the arms in front of the chest, the actor returns to an intermediate position where the arms are neither straight down, nor wide open, so that two movemes are needed to explain that configuration).

5.4 Modeling the Dynamics

Although the approach we have presented in the previous section shows interesting segmentation results, the idea of independently assigning individual frames to movemes fails to represent subtle aspects such as the dynamic of the motion. Since our goal is to eventually be able to recognize and understand the complex and rich behaviors that a human body is capable of, we must be careful not to discard essential information when choosing our model.

If we observe people involved in everyday activities, we can summarize their motion as the trajectories of their body parts in time. It is true that there are times in which it may seem superfluous to know the exact temporal evolution of a body part. Other times, however, when an arm reaches to a nearby location to grab an object, for example, it traverses the same set of poses as an arm punching an obstacle. The different semantic of these two movements is encoded by their dynamics. Accurately representing such information is important, yet challenging, since the dynamics are complex, non-linear and possibly time-varying.

Modeling the dynamics has additional and important benefits. For example, when doing visual tracking the prediction capabilities of a dynamical model can help in coping with occlusions and errors in the observations. Applications in the computer graphics community, such as animation and rendering, are also greatly improved by more realistic synthetic motion.

Although several attempts have been made in recent years to represent some form of dynamical information, the field is still relatively new and evolving. In the reminder of this chapter we present our approach to learning models to represent the dynamics of human motion. Our work is based on an existing modeling technique known as the *switching linear dynamical system (SLDS)*. A comprehesive overview of the different algorithms for learning and inference in switching dynamical systems can be found in [GH96, GH98, Gha98, Mur98, BK98, GH00, Mur02, ORBD05]. We refer to [Bar78, SS82, SS91, BL93, Kim94] for earlier attempts in tracking and, more in generally, for modeling signals based on SLDS. Several applications of dynamical models in biology and vision can be seen in [PR00, Roh97, Bre97, NBIR00, DMP02, DMP03a, DMP03b].

5.4.1 Switching Linear Dynamical Systems

One of the most successful approach to dynamics modeling is given by the switching linear dynamical system (SLDS), also known as the jump Markov system (JMS), or switching Kalman filter (SKF). Murphy [Mur98] provides an excellent introduction to both the inference and learning tasks with SLDS, to which we refer. Here, we summarize[1] Murphy's derivation following his terminology and notation, while making the due adaptations to our specific problem.

In their simplest form, SLDS can be described as a pool of linear dynamical systems (LDS), whose parameters have been set. At any given time a selector or *switch* variable determines which one of the linear dynamics is to be used. The switch variable itself evolves in time according to a first-order Markov dynamic.

[1]The work in [Mur98] is ©1998 Kevin P. Murphy and Compaq Computer Corporation, Cambridge Research Laboratory, Cambridge, MA 02139 USA. The partial reproduction and adaptation in this thesis is done with permission, for nonprofit educational purposes; all other uses require a license and payment of fee to the copyright owners.

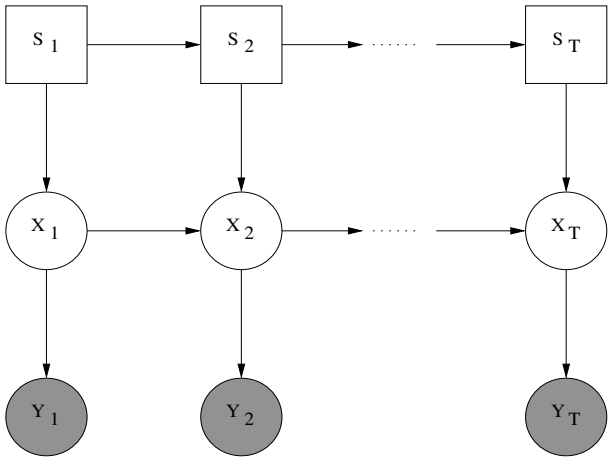


Figure 5.10: Switching Linear Dynamical System. The switch variable s_t evolves in time according to a first-order Markov chain. For a given choice of the switch, a corresponding dynamic is imposed on the evolution of $x_{t-1} \rightarrow x_t$. Alternatives to the switching dynamic configurations, such as switching observations, are also possible.

Figure 5.10 shows a graphical model describing a time-unrolled SLDS, going from $t = 1$ to $t = T$. This is called the switching-dynamics form and is the focus of our attention. Alternative switching mechanisms are possible. For example, in the switching-observations form, the switch variable controls the production of the observation y_t from the state x_t, allowing for a multi-modal observation process.

Unfortunately, most physical processes exhibit complex non-linear dynamics which can at best be approximated locally by a linear system. Since an SLDS possesses a number of interchangeable dynamics, it can be seen as a piecewise linear approximation of a non-linear process. At the core of the SLDS are the linear dynamics and the common hypothesis of Gaussian noise. This can be represented as follows:

$$x_t = Ax_{t-1} + B + v_t \tag{5.2}$$

$$y_t = Cx_t + w_t \tag{5.3}$$

where x_t is the hidden state of the system, while y_t represents the observation at time t. $v_t \sim \mathcal{N}(0, Q)$ and $w_t \sim \mathcal{N}(0, R)$ are the (independent) state noise and observation noise, respectively, and model the uncertainty in the state evolution and measurement process.

Inference in this model is efficient and simple. Both the Kalman *filter* and the Rauch-Tung-Strieber *smoother* are well known on-line procedures which allow us to compute the posterior on the state $P(x_t|y_{1:t})$ given past observations, or given all the available data $P(x_t|y_{1:T})$.

The switching variable s_t has a Markovian dynamic, that is, s_1 has an initial distribution π, while the evolution is governed by the transition matrix Z.

The full SLDS model can be written as follows:

$$\begin{aligned} s_t &\sim Z(s_{t-1}, \cdot) \\ x_t &= A(s_t)x_{t-1} + B(s_t) + v_t \qquad & v_t &\sim \mathcal{N}(0, Q(s_t)) \\ y_t &= C(s_t)x_t + w_t & w_t &\sim \mathcal{N}(0, R(s_t)) \end{aligned} \tag{5.4}$$

which is the most general form of SLDS, since the evolution of both state and observation depend on the switch.

5.4.1.1 Belief Approximation

If a LDS admits efficient inference, the introduction of the switching state brings an additional layer of complexity. To see this, let us start with the initial distribution $f(x_1)$, which is a mixture of Gaussians with one component for each value of the switch s_1. Each component in the mixture has to run through the dynamics equations. Since there is one dynamic/observation equation pair for each value of s_2, we obtain a resulting mixture with M^2 components. After t iteration, the mixture $p(x_t|y_{1:t})$ has grown exponentially to M^t

components.

Unfortunately, we are left with little choice but to approximate. [Mur98] offers a rather complete review and characterization of the various approximations appeared in the literature. In our work, we focus on the idea of [BL93], which is known as the second-order generalized pseudo-Bayesian (GBP2) algorithm[2].

In general, the GBP(r) approximation of order r consists of collapsing (at time t) a larger mixture into one with M^{r-1} components. When $r = 2$, we take $f(x_{t-1}|y_{1:t-1})$, which is an M-dimensional mixture, and we run it through our multi-modal filtering. We then approximate the resulting M^2 components by moment matching[3], returning to an M-components mixture for $f(x_t|y_{1:t})$.

5.4.1.2 Inference in SLDS

In this section we show how to do inference (both filtering and smoothing) in SLDS. Since this is now a standard process, we report the derivation directly from [Mur98], with the appropriate modifications to take into account the additional input matrix $B(s_t)$. We start by introducing their notation:

$$
\begin{aligned}
x_{t|\tau}^{i(j)} &= \mathbb{E}\left[x_t|y_{1:\tau}, s_{t-1} = i, s_t = j\right] \\
x_{t|\tau}^{(j)k} &= \mathbb{E}\left[x_t|y_{1:\tau}, s_t = j, s_{t+1} = k\right] \\
x_{t|\tau}^{j} &= \mathbb{E}\left[x_t|y_{1:\tau}, s_t = j\right].
\end{aligned}
$$

[2]GBP is a particular instance of a more general approximating technique known as assumed density filtering (ADF). In ADF, a belief is represented with a density which is member of a particular family. Any time an update is performed that results in a belief outside of the desired family, the belief is approximated with a density from the family.

[3][Lau96] has shown that moment matching results in the optimal approximation with respect to the K-L divergency.

If $\tau = t$, the above represent the filtered statistics; when $\tau > t$, they are called smoothed statistics; finally, if $\tau < t$, we talk about predicted statistics. The superscript inside the brackets is the value of the switch node at the time specified by the subscript t. The superscript to the left and to the right are instead the values of s_{t-1} and s_{t+1}, respectively. We also define

$$\begin{aligned}
V_{t|\tau}^{j} &= \mathrm{Cov}\left[x_t | y_{1:\tau}, s_t = j\right] \\
V_{t,t-1|\tau}^{j} &= \mathrm{Cov}\left[x_t, x_{t-1} | y_{1:\tau}, s_t = j\right] \\
V_{t,t-1|\tau}^{(i)j} &= \mathrm{Cov}\left[x_t, x_{t-1} | y_{1:\tau}, s_{t-1} = i, s_t = j\right] \\
M_{t-1,t|\tau}(i,j) &= f(s_{t-1} = i, s_t = j | y_{1:\tau}) \\
M_{t|\tau}(j) &= f(s_t = j | y_{1:\tau}) \\
L_t^j &= f(y_t | y_{1:t-1}, s_t = j)
\end{aligned}$$

where L_t^j is the likelihood of the innovation at time t, given that the current model is j.

Filtered Estimates

In order to produce the filtered estimates of the hiddent state x_t and switches s_t, given the observations $y_{1:t}$ up to time t, we perform the following steps in sequence:

$$
\begin{aligned}
(x_{t|t}^{i(j)}, V_{t|t}^{i(j)}, V_{t,t-1|t}^{(i)j}, L_t^{i(j)}) &= \text{Filter}(x_{t-1|t-1}^i, V_{t-1|t-1}^i, y_t; A_j, B_j, C_j, Q_j, R_j) \\
M_{t-1,t|t}(i,j) &= \frac{L_t(i,j) Z(i,j) M_{t-1|t-1}(i)}{\sum_i \sum_j L_t(i,j) Z(i,j) M_{t-1|t-1}(i)} \\
M_{t|t}(j) &= \sum_i M_{t-1,t|t}(i,j) \\
W_t^{i|j} &= f(s_{t-1} = i | s_t = j, y_{1:t}) = M_{t-1,t|t}(i,j) / M_{t|t}(j) \\
(x_{t|t}^j, V_{t|t}^j) &= \text{Collapse}(x_{t|t}^{i(j)}, V_{t|t}^{i(j)}, W_t^{i|j}).
\end{aligned}
$$

At the first iteration we use the following initial conditions:

$$
\begin{aligned}
x_{1|0}^j &\triangleq \mathbb{E}[x_1 | s_1 = j] = \mu^j \\
V_{1|0}^j &\triangleq \text{Cov}[x_1 | s_1 = j] = \Sigma^j \\
M_{0|0} &\triangleq \pi.
\end{aligned}
$$

The filter and collapse operators are defined as follows:

Filter :

$$
(x_{t|t}, V_{t|t}, V_{t,t-1|t}, L_t) = \text{Filter}(x_{t-1|t-1}, V_{t-1|t-1}, y_t; A, B, C, Q, R)
$$

First, we compute the predicted mean and variance.

$$
\begin{aligned}
x_{t|t-1} &= A x_{t-1|t-1} + B \\
V_{t|t-1} &= A V_{t-1|t-1} A' + Q
\end{aligned}
$$

Then we compute the prediction error (called "innovation"), the variance of the error,

the Kalman gain matrix, and the likelihood of the current observation.

$$\begin{aligned} e_t &= y_t - Cx_{t|t-1} \\ S_t &= CV_{t|t-1}C' + R \\ K_t &= V_{t|t-1}C'S_t^{-1} \\ L_t &= \mathcal{N}(e_t; 0, S_t) \end{aligned}$$

Finally, we update the estimates of mean, variance, and cross variance.

$$\begin{aligned} x_{t|t} &= x_{t|t-1} + Ke_t \\ V_{t|t} &= (I - K_tC)V_{t|t-1} = V_{t|t-1} - K_tS_tK_t' \\ V_{t,t-1|t} &= (I - K_tC)AV_{t-1|t-1} \end{aligned}$$

CollapseCross :

Given two random variables x, y with conditional means $\mu_x^j = \mathbb{E}[x|s = j]$, $\mu_y^j = \mathbb{E}[y|s = j]$, cross variance $V_{x,y}^j = \mathrm{Cov}[x, y|s = j]$, and mixing coefficient $P^j = f_s(j)$, the CollapseCross operator is defined as follows:

$$(\mu_x, \mu_y, V_{x,y}) = \mathrm{CollapseCross}(\mu_x^j, \mu_y^j, V_{x,y}^j, P^j)$$

where

$$\begin{aligned}
\mu_x &\triangleq \sum_j P^j \mu_x^j \\
\mu_y &\triangleq \sum_j P^j \mu_y^j \\
V_{x,y} &\triangleq \sum_j P^j V_{x,y}^j + \sum_j P^j (\mu_x^j - \mu_x)(\mu_y^j - \mu_y)'.
\end{aligned}$$

Collapse :

$$\text{Collapse}(\mu_x^j, V_x^j, P^j) \triangleq \text{CollapseCross}(\mu_x^j, \mu_y^j, V_{x,y}^j, P^j).$$

Smoothed Estimates

In order to produce the smoothed estimates of the hidden state x_t and switches s_t, given all the observations available $y_{1:T}$, we perform the following steps in sequence:

$$\begin{aligned}
(x_{t|T}^{(j)k}, V_{t|T}^{(j)k}, V_{t+1,t|T}^{j(k)}) &= \text{Smooth}(x_{t+1|T}^k, V_{t+1|T}^k, x_{t|t}^j, V_{t|t}^j, V_{t+1|t+1}^k, V_{t+1,t|t+1}^{j(k)}; A_k, B_k, Q_k) \\
U_t^{j|k} &= f(S_t = j | s_{t+1} = k, y_{1:T}) \approx \frac{M_{t|t}(j) Z(j,k)}{\sum_b M_{t|t}(b) Z(b,k)} \qquad * \\
M_{t,t+1|T}(j,k) &= U_t^{j|k} M_{t+1|T}(k) \\
M_{t|T}(j) &= \sum_k M_{t,t+1|T}(j,k) \\
W_t^{k|j} &= f(s_{t+1} = k | s_t = j, y_{1:T}) = M_{t,t+1|T}(j,k) / M_{t|T}(j)
\end{aligned}$$

$$
\begin{aligned}
(x^j_{t|T}, V^j_{t|T}) &= \text{Collapse}(x^{(j)k}_{t|T}, V^{(j)k}_{t|T}, W^{k|j}_t) \\
(x_{t|T}, V_{t|T}) &= \text{Collapse}(x^j_{t|T}, V^j_{t|T}, M_{t|T}(j)) \\
x^{j(k)}_{t+1|T} &= \mathbb{E}[x_{t+1}|y_{1:T}, s_{t+1} = k, S_t = j] \approx x^k_{t+1|T} \\
V^k_{t+1,t|T} &= \text{CollapseCross}(x^{j(k)}_{t+1|T}, x^{(j)k}_{t|T}, V^{j(k)}_{t+1,t|T}, U^{j|k}_t) \\
x^{()k}_{t|T} &= \mathbb{E}[x_t|y_{1:T}, s_{t+1} = k] = \sum_j x^{(j)k}_{t|T} U^{j|k}_t \\
V_{t+1,t|T} &= \text{CollapseCross}(x^k_{t+1|T}, x^{()k}_{t|T}, V^k_{t+t,t|T}, M_{t+1|T}(k)).
\end{aligned}
$$

The line marked $*$ is a standard approximation [Kim94], which is due to the fact that S_t is not independent of the future evidence $y_{t+1} \dots y_T$, even if S_{t+1} is given.

The smooth operator is defined as follows:

Smooth :

$$(x_{t|T}, V_{t|T}, V_{t+1,t|T}) = \text{Smooth}(x_{t+1|T}, V_{t+1|T}, x_{t|t}, V_{t|t}, V_{t+1|t+1}, V_{t+1,t|t+1}; A, Q)`$$

First, we compute the predicted mean and variance.

$$
\begin{aligned}
x_{t+1|t} &= Ax_{t|t} + B \\
V_{t+1|t} &= AV_{t|t}A' + Q
\end{aligned}
$$

Then we compute the smoother gain matrix.

$$J_t = V_{t|t}A'V^{-1}_{t+1|t}$$

Finally, we update the estimates of mean, variance, and cross variance.

$$\begin{aligned} x_{t|T} &= x_{t|t} + J_t(x_{t+1|T} - x_{t+1|t}) \\ V_{t|T} &= V_{t|t} + J_t(V_{t+1|T} - V_{t+1|t})J_t' \\ V_{t+1,t|T} &= V_{t+1|T}V_{t+1|t+1}^{-1}V_{t+1,t|t+1}. \end{aligned}$$

5.4.1.3 Learning with Expectation Maximization

In this section we show how to learn the parameters of an SLDS from data. We assume that a number N of i.i.d. observation sequences $y_{1:T_l}^l$ is available and we wish to optimize their likelihood with respect to the model parameters. Once again, we borrow Murphy's notation and derivations from [Mur98], with the appropriate modifications to take into account the additional input matrix $B(s_t)$.

The complete-data log-likelihood for a single sequence is given by

$$\begin{aligned} L &= \log f(x,s,y) = \log f(x_{1:T}, s_{1:T}, y_{1:T}) \\ &= -\frac{1}{2}\sum_{t=1}^{T}([y_t - C_t x_t]' R_t^{-1}[y_t - C_t x_t]) - \frac{1}{2}\sum_{t=1}^{T}\log|R_t| \\ &\quad -\frac{1}{2}\sum_{t=1}^{T}([x_t - A_t x_{t-1} - B_t]' Q_t^{-1}[x_t - A_t x_{t-1} - B_t]) - \frac{1}{2}\sum_{t=2}^{T}\log|Q_t| \\ &\quad -\frac{1}{2}[x_1 - \mu_1]'\Sigma_1^{-1}[x_1 - \mu_1] - \frac{1}{2}\log|\Sigma_1| - \frac{T(n+m)}{2}\log 2\pi \\ &\quad +\log\pi_1 + \sum_{t=2}^{T}\log Z(s_{t-1}, s_t). \end{aligned}$$

The expected-complete log-likelihood which we wish to maximize is

$$
\begin{aligned}
\hat{L} &= \mathbb{E}_{f_{sx|y}}[L] \\
&= \mathbb{E}_{f_{s|y}}\left[\mathbb{E}_{f_{x|sy}}[L]\right] \\
&\approx \mathbb{E}_{f_{s|y}}\left[\mathbb{E}_{f_{x|y}}[L]\right] \qquad * \\
&= \sum_{s_1} \cdots \sum_{s_T} f(s|y)\left[\hat{\mathbb{E}}[L]\right] \\
&= \sum_{t=2}^{T} \sum_{s_t} \left(\sum_{\{s_r, r \neq t\}} f(s|y) \right) \left[\hat{\mathbb{E}}[\log f(x_t|x_{t-1}, s_t)] + \ldots\right] \\
&= \sum_{t=2}^{T} \sum_{s_t} f(s_t|y_{1:T}) \hat{\mathbb{E}}[\log f(x_t|x_{t-1}, s_t)] + \ldots \\
&= \sum_{t=2}^{T} \sum_{s_t} W_t^{s_t} \hat{\mathbb{E}}[\log f(x_t|x_{t-1}, s_t)] + \ldots
\end{aligned}
$$

where $\hat{\mathbb{E}}[\cdot] \triangleq \mathbb{E}[\cdot|y_{t:T}]$. The approximation in $*$ arises because we compute the expectation with respect to $f(x_t|y_{1:T})$, rather than $f(x_t|s_{1:T}, y_{1:T})$, since the latter is an exponentially large mixture of components (one for each segmentation induced by $s_{1:T}$).

To simplify the notation in the maximizations we are about to derive, we define the following quantities, which are all computed during the inference step:

$$
\begin{aligned}
W_t^j &\triangleq f_{s_t|y}(j|y_{1:T}) \\
\hat{x}_t &\triangleq \hat{\mathbb{E}}[x_t] \\
P_t &\triangleq \hat{\mathbb{E}}[x_t x_t'] = V_{t|T} + x_{t|T} x_{t|T}' \\
P_{t,t-1} &\triangleq \hat{\mathbb{E}}[x_t x_{t-1}'] = V_{t,t-1|T} + x_{t|T} x_{t-1|T}'
\end{aligned}
$$

System Matrices :

In order for us to optimize with respect to A_j and B_j simultaneously, we introduce the vector $z_t = [x_t', 1]'$ and the matrix $F = [A_j, B_j]$. The above definitions then become

$$\begin{aligned} \hat{z}_t &\triangleq \hat{\mathbb{E}}[z_t] \\ U_t &\triangleq \hat{\mathbb{E}}[z_t z_t'] = \begin{bmatrix} P_t & \hat{z}_t \\ \hat{z}_t' & 1 \end{bmatrix} \\ U_{t,t-1} &\triangleq \hat{\mathbb{E}}[z_t z_{t-1}'] = \begin{bmatrix} P_{t,t-1} & \hat{z}_t \\ \hat{z}_{t-1}' & 1 \end{bmatrix}. \end{aligned}$$

We can now maximize $\hat{L}$ with respect to F obtaining

$$\begin{aligned} 0 = \frac{\partial}{\partial F_j}\hat{L} &= -\frac{1}{2}\sum_{t=2}^{T} W_t^j \hat{\mathbb{E}}[2Q_j^{-1}(x_t - F_j z_{t-1})] \\ &= -\sum_{t=1}^{T} W_t^j Q_j^{-1} P_{t,t-1} + \sum_{t=2}^{T} W_t^j Q_j^{-1} F_j U_{t-1}. \end{aligned}$$

Hence

$$F_j = \left(\sum_{t=2}^{T} W_t^j U_{t,t-1}\right)\left(\sum_{t=2}^{T} W_t^j U_{t-1}\right)^{-1}.$$

System Noise Covariance :

$$\begin{aligned} 0 = \frac{\partial}{\partial Q_j^{-1}}\hat{L} &= -\frac{1}{2}\sum_{t=2}^{T} W_t^j \hat{\mathbb{E}}[(x_t - F_j z_{t-1})(x_t - F_j z_{t-1})] + \frac{1}{2}\sum_{t=2}^{T} W_t^j Q_j \\ &= -\sum_{t=1}^{T} W_t^j [U_t - F_j U'_{t,t-1} - U_{t,t-1}F'_j + F_j U_{t-1} F'_j] + \frac{1}{2}\sum_{t=2}^{T} W_t^j Q_j. \end{aligned}$$

Using the new value of F_j and the fact that U_t is symmetric, we have

$$\begin{aligned} F_j \left(\sum_{t=2}^{T} W_t^j U_{t-1}\right) F'_j &= \left(\sum_{t=2}^{T} W_t^j U_{t,t-1}\right)\left(\sum_{t=2}^{T} W_t^j U_{t-1}\right)^{-1}\left(\sum_{t=2}^{T} W_t^j U'_{t,t-1}\right) \\ &= F_j \sum_{t=2}^{T} W_t^j U'_{t,t-1} = \left(\sum_{t=2}^{T} W_t^j U_{t,t-1}\right) F'_j. \end{aligned}$$

Hence

$$Q_j = \left(\frac{1}{\sum_{t=2}^{T} W_t^j}\right)\left(\sum_{t=2}^{T} W_t^j U_t - F_j \sum_{t=2}^{T} W_t^j U'_{t,t-1}\right).$$

Observation Matrices :

$$0 = \frac{\partial}{\partial C_j}\hat{L} = -\frac{1}{2}\sum_{t=1}^{T} W_t^j \hat{\mathbb{E}}[2R_j^{-1}(-C_j x_t + y_t)].$$

Hence

$$C_j = \left(\sum_{t=1}^{T} W_t^j y_t \hat{x}'_t\right)\left(\sum_{t=1}^{T} W_t^j P_t\right)^{-1}.$$

Observation Noise Covariance :

$$0 = \frac{\partial}{\partial R_j^{-1}}\hat{L} = \sum_{t=1}^{T} \hat{\mathbb{E}}[W_t^j \frac{1}{2}(y_t y'_t - 2C_j x_t y'_t + C_j x_t x'_t X'_j)] + \frac{1}{2} R_j \sum_{t=1}^{T} W_t^j.$$

Using the newly computed estimate for C_j, we have

$$\left(\sum_{t=1}^{T} W_t^j P_t\right) C_j' = \sum_{t=1}^{T} W_t^j \hat{x}_t y_t' \triangleq Z$$

so

$$\frac{\partial}{\partial R_j^{-1}} \hat{L} = \frac{1}{2} \sum_{t=1}^{T} W_t^j y_t y_t' - 2C_j Z + C_j Z + \frac{1}{2} R_j \sum_{t=1}^{T} W_t^j$$

and hence

$$R_j = \left(\frac{1}{\sum_{t=1}^{T} W_t^j}\right) \sum_{t=1}^{T} W_t^j (y_t y_t' - C_j \hat{(x)}_t y_t').$$

Initial Mean and Covariance, Switch Probability and Transition Matrix :

These are the standard derivations for a mixture of Gaussians and for an HMM, respectively. They can be found in most probability/learning books, such as [Bis07].

5.4.2 The Limb Model

Now that we have reviewed how learning and inference is done in switching linear dynamical system models, we discuss their application to the problem of moveme discovery.

Due to the GPB2 approximation, inference and learning in SLDS has a complexity which is quadratic in the number of discrete switches. This restricts the number of linear components we are willing to handle in our model. On the other hand, the physical process we are trying to model is capable of complex dynamics, which might require the use of a large number of components. Furthermore, as the number of parts modeled increases, the complexity of the systems might, in principle, grow exponentially.

In our earlier work, we have already noted the benefits of modeling human movements by decomposing the body into a number of smaller parts, rather than modelling it as a

whole. Here, we propose to model individual trajectories of a few points on the body, which results in a simpler and more tractable problem. Since we are primarily interested in the motion of the body, and not its appearence, we exclude head and torso and focus on the four limbs instead.

For each limb, we select three parts which coincide with the joints (For an arm: shoulder, elbow, and wrist. For a leg: hip, knee, and ankle). The trajectory of each part is described by a pair of SLDS, one for the horizontal and one for the vertical motion. The equations describing the evolution of a part are as follows:

$$\begin{aligned} s_t &\sim Z(s_{t-1}, \cdot) \\ x_t &= A(s_t)x_{t-1} + B(s_t) + v_t \qquad & v_t &\sim \mathcal{N}(0, Q(s_t)) \\ y_t &= C(s_t)x_t + w_t & w_t &\sim \mathcal{N}(0, R(s_t)). \end{aligned} \tag{5.5}$$

After experimenting with models of the first and second order, with constant velocity and constant acceleration, and with different types of coupling and constrains, we have settled for a simple first-order SLDS with scalar state and constant velocity. This implies that the system matrix $A(s_t) = 1$ is fixed. The observations matrix $C(s_t) = 1$ is also fixed, since we are able to observe a noisy version of the state directly, and our data is hand labeled and complete. The parameter $B(s_t)$ represents the constant velocity. The EM procedure of the previous section is used to learn estimates for $B(s_t)$, $R(s_t)$, and $Q(s_t)$, as well as the parameters π and Z of the switch's dynamic. To reduce the number of models needed, and to simplify the dynamics that we represent, we compute the relative position of a part with respect to its immediate predecessor in the body. For example, we model the wrist relative to the elbow, and the elbow relative to the shoulder, and so on.

We allow five components to be learned for each coordinate, which seems to be suffi-

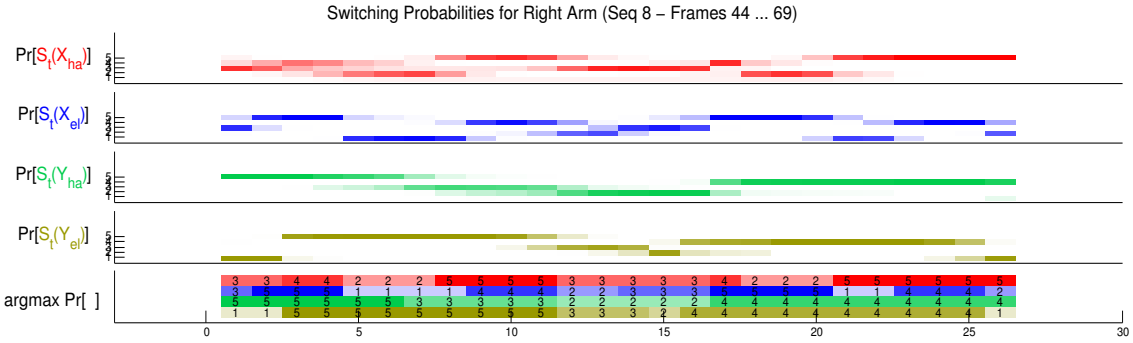


Figure 5.11: Switching Probabilities. [Top 4 Rows] For each of the four coordinates in the right arm we show the posterior $f(s_t|y_{1:T})$ of the switch variable s_t. [Bottom Row] The most probable switches are pooled together. The intensity/number indicates which of the 5 switches was chosen.

cient for the trajectories we observe. Had we modeled multiple parts within the same SLDS instead of single coordinates, we would have probably needed tens or even hundreds of different components to represent the same variety of movements. Clearly, the computational saving comes at the price: we loose the coupling between variables, which may provide some information about the motion.

Once the parameters of each SLDS have been estimated, we combine the individual representations into a single model, which we call the *limb model*. More specifically, for each one of the four coordinates within a limb, we compute the most likely configuration of the switch variable s_t, and then join them together into a four-dimensional *descriptor* representing the instantaneous motion of the limb. For example, the right arm encoding would be computed as follows:

$$c_t \triangleq \begin{bmatrix} \arg\max_{s_t} f(s_t(X_{\mathbf{ha}})|y_{1:T}) \\ \arg\max_{s_t} f(s_t(Y_{\mathbf{ha}})|y_{1:T}) \\ \arg\max_{s_t} f(s_t(X_{\mathbf{el}})|y_{1:T}) \\ \arg\max_{s_t} f(s_t(Y_{\mathbf{el}})|y_{1:T}) \end{bmatrix} \tag{5.6}$$

where $(X_{\mathbf{ha}}, Y_{\mathbf{ha}})$ and $(X_{\mathbf{el}}, Y_{\mathbf{el}})$ are the X and Y coordinates of the hand and elbow (relative to the elbow and shoulder), respectively. Figure 5.11 shows the posterior of the switch variables in a few sample frames, and the vectors obtained by aggregating the most probable value for each coordinate.

5.4.3 Motion Codewords

The descriptor we have just computed carries limited information on its own, since the temporal extent of the movement it represents, is very limited. As we mentioned at the beginning of the chapter, we believe that the atomic components of motion range in duration from a few frames to a second or so. We thus need to consider short sequences of descriptors, which we call *motion codewords*.

Codewords can be thought of as exemplars, or signatures of a characteristic motion. Given two repetitions of the same action, we expect their codewords to be similar in length and descriptors, while different motions and/or dynamics would produce different codewords.

Unfortunately, we are given a continuous video sequence (or its encoding with the descriptors of (5.6)). Since we desire a codeword-based representation of the sequence, we have to determine a suitable segmentation. The first step is clearly to enforce constraints on the length of each segment. Next, we make sure that two portions of the video (or, rather, two sequences of descriptors) which represent the same motion are segmented into similar codewords. Notice how the concept of distance between two codewords is not meant as a descriptor-wise matching: some of the descriptor within the codeword may be mismatched, and some shifting should be allowed due to the (possibly) different length of the codewords. Finally, we wish for our decomposition to represent the totality of the sequence, since no

movements should go unexplained. These competing requirements make the problem of segmenting descriptor sequences into codewords quite challenging.

5.4.3.1 Motif Discovery

A simpler problem than the one we are facing, is the search for approximate copies of a prototype sub-sequence, which is embedded at unknown locations in a longer sequence. This is a rather common problem which arises in several other disciplines, such as medicine, meteorology, finance, and robotics.

The biology community has been investigating the problem of discovering relevant subsequences of DNA sequences for quite some time. Since a gene (or group of genes) in the DNA encodes for various functionality in the organism, the hope is to determine which portions of the chain are responsible for diseases or malfunctions. The implications are clearly of great importance since this knowledge could, for example, allow the early detection or even prevention of life-threatening conditions, at a time when such events would normally go undetected. The search for known subsequences within a corpus of longer sequences is generally called *motif detection*. When the subsequence responsible for a feature of interest is instead unknown, the term *motif discovery* is more appropriate, since one can only rely on the fact that copies of the subsequence are embedded somewhere in each sequence of the corpus. Clearly, it is known that the set of sequences in the corpus comes from a population of organisms that exhibit the feature of interest.

A significant amount of work on *pattern discovery* and *detection* has been applied to problems in finance, and more generally in economics. There the goal is to identify patterns of prices, or some set of derived indicators or indexes, which reliably predict phenomenons of interest, such as sudden spikes in pricing or significant changes in the equilibrium of a

market or economy.

Although both the field of application and the underlying interpretation are quite different, the fact that these sets of problems all share several common features at their core has guided the different communities towards the development of similar techniques and solutions. Among the most influential results are probabilistic approaches based on profile-hidden Markov models (PHMM), as well as some heuristic-based methods with interesting computational properties.

5.4.3.2 Energy-Based Segmentation

Notwithstanding the large amount of interest and effort from different fields of research, the task of simultaneously segmenting a sequence into codewords and grouping them based on their mutual similarity, remains an open problem. In this work we present a suboptimal solution which, however, produces appealing results when applied to our particular problem.

The empirical observation we make is that, when the motion of a limb reaches a stationary point, this is usually a good candidate for a segmental point. More specifically, we compute an energy value for each instant in time. The energy is based on the standard deviation (in a small temporal neighborhood) of the position of each coordinate in the limb. Local minima of the energy seem to correlate well with approximate stationarity or visually meaningful (to a human) changes in the motion of the limb.

The segmental points are determined based on the energy and subject to a minimum and maximum length of the segment, with the duration-constraint taking priority over the energy-proposed segmental points.

Figure 5.12 shows an example of the limb's trajectories and the corresponding energy. The vertical lines define the segmentation points.

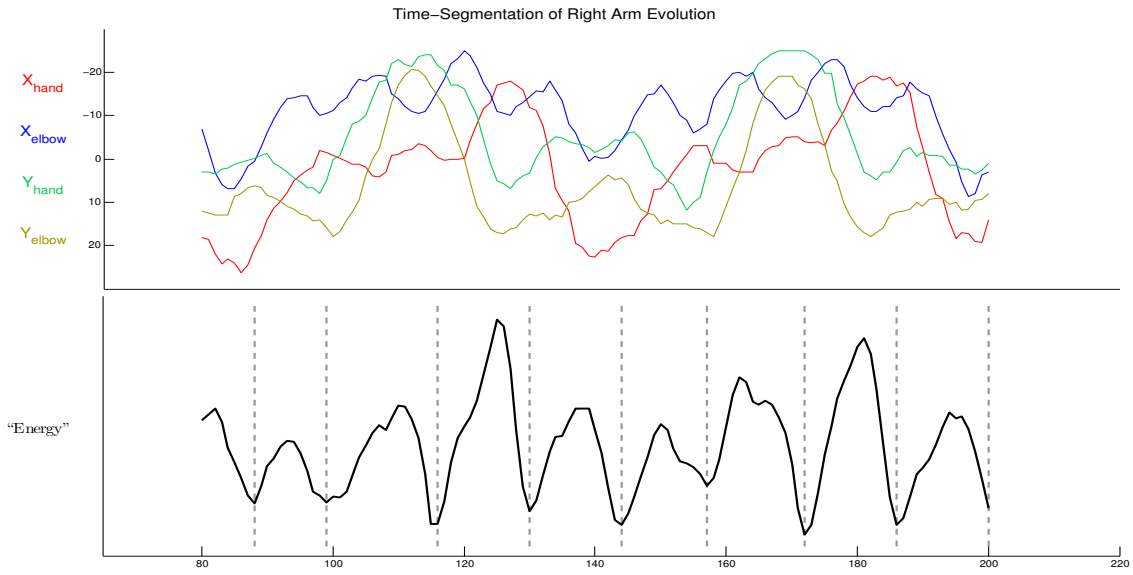


Figure 5.12: Energy-Based Segmentation. [Top] We show sample trajectories of the four coordinates for the right arm. [Bottom] The energy is computed by averaging the standard deviation of the coordinates in a small temporal neighborhood. The dashed vertical lines demarcate the segments' boundaries.

5.4.4 Clustering Variable-Length Codewords into Movemes

Now that we have represented the motion of a limb as a sequence of codewords, we seek to group codewords into clusters. As mentioned in the previous section, we would like the portions of a video containing the same motion to be represented in the same way. Since in practice similar (but not identical) codewords will be associated with repetitions of the same action, grouping will provide a single identifier for each set of self-similar motions: we call such identifier a *moveme*[4]

5.4.4.1 Pairwise Distance Measure

The first step to clustering the codewords is to compute their affinity matrix $[A(a,b)]$. For each pair of codewords w_a and w_b the entry $A(a,b) = d(w_a, w_b)$ measures the similarity

[4]In human motion recognition, the term "*moveme*" is meant as an atomic movement and is dervied by analogy from the term "phoneme". It was first proposed by Bregler [Bre97]

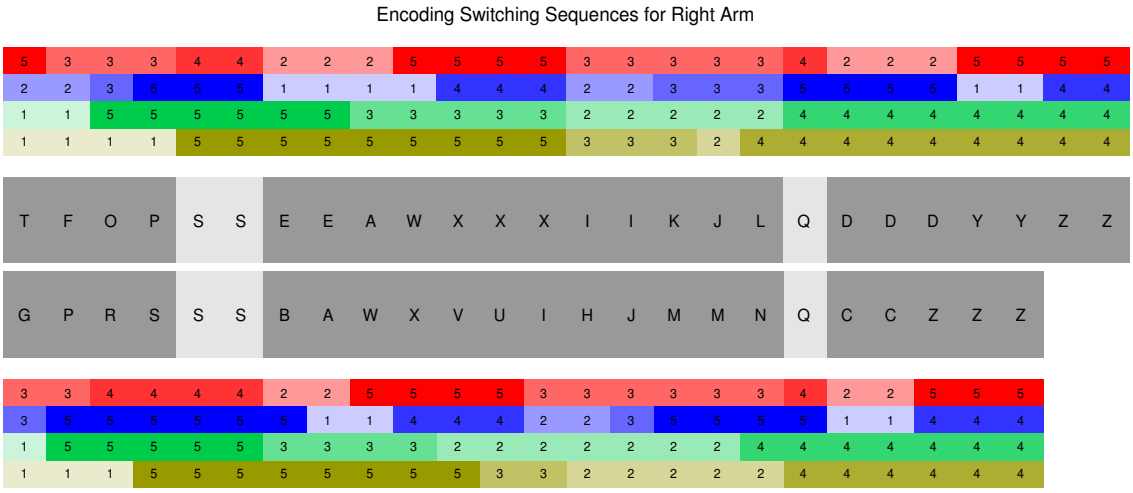


Figure 5.13: Letter-Based Representation of Codewords. [Top and Bottom] We show two codewords representing identical (to a human) movements appearing in different parts of a video sequence. The numbers within each descriptor were obtained according to the (5.6) and indicate the dynamics followed by each coordinate in the limb. [Center Rows] We uniquely map a descriptor vector to a letter of the alphabet. The resulting letter-based representation is shown.

(or affinity) of the pair. This is not a trivial problem, since the codewords have different lengths. Additionally, we are not interested in a one-to-one matching of the descriptors within the codewords, since a small shifting within one codeword could erroneously result in the codewords being very dissimilar. To illustrate the problem, Figure 5.13 shows two codewords representing identical (to a human) movements appearing in different parts of a video sequence. The numbers within each descriptor were obtained according to the (5.6) and indicate the dynamics followed by each coordinate in the limb. To simplify the visualization, we can pretend that the descriptors appearing in either of the codewords are the only ones which are possible. Since this is a rather small set (compared to the $5^4 = 625$ possible descriptors) we can uniquely map a descriptor vector to a letter of the alphabet. The central rows of the figure show the new letter-based representation for the two codewords. Although a quick glance at the codewords reveals a high degree of similarity, the highlighted one-to-one correspondence of letters does not convey a great level of affinity.

A similar problem is encountered in biology, when performing Motif discovery and detection. There segments of DNA or proteins that have undergone random mutations result in pertinent nucleotides being missing or changed, or spurious ones wrongly inserted.

Similarly, spell-checking software, in an attempt to propose useful substitutions, compares a misspelled word against entries in a dictionary. The similarity with an entry is given by the amount of editing, measured as the number of insertions, deletions, or replacements of a character, that are necessary to render the two strings identical. This similarity is known in the field as the *edit distance*[5].

Luckily, determining the optimal alignment of symbols within two sequences can be accomplished by an efficient algorithm based on dynamic programming. The iterative procedure, known as the Smith-Waterman algorithm [SW81] computes the alignments between all possible sub-sequences of the strings. To score a specific solution, a set of costs are provided to the algorithm as parameters. Typically, insertions and deletions reduce the score by an amount that grows linearly with the size of the gap. A table, containing the penalty incurred when replacing a symbol with another one, is also provided to the algorithm. The cost of substitution, which is problem-dependent, allows for a lesser penalty when, for example, the interchanged symbols are representative of different but sufficiently similar underlying structure.

The algorithm iteratively fills a $\text{len}(w_a) \times \text{len}(w_b)$ matrix $[S_{ab}(i,j)]$ of scores, with the symbols in the two strings placed as headers of columns and rows (that is, there is one column for each symbol in the first codeword w_a, and one row for each symbol in the second codeword w_b). The process selects one entry at a time, scanning the table from left to right, top to bottom. Figure 5.14 shows the table setup for the sample codewords of

[5]The specific way of measuring the similarity as "the *number* of editing operations" was first proposed in [Lev66] and named *Levenshtein distance*. This is just one of many possible way to measure edit distance.

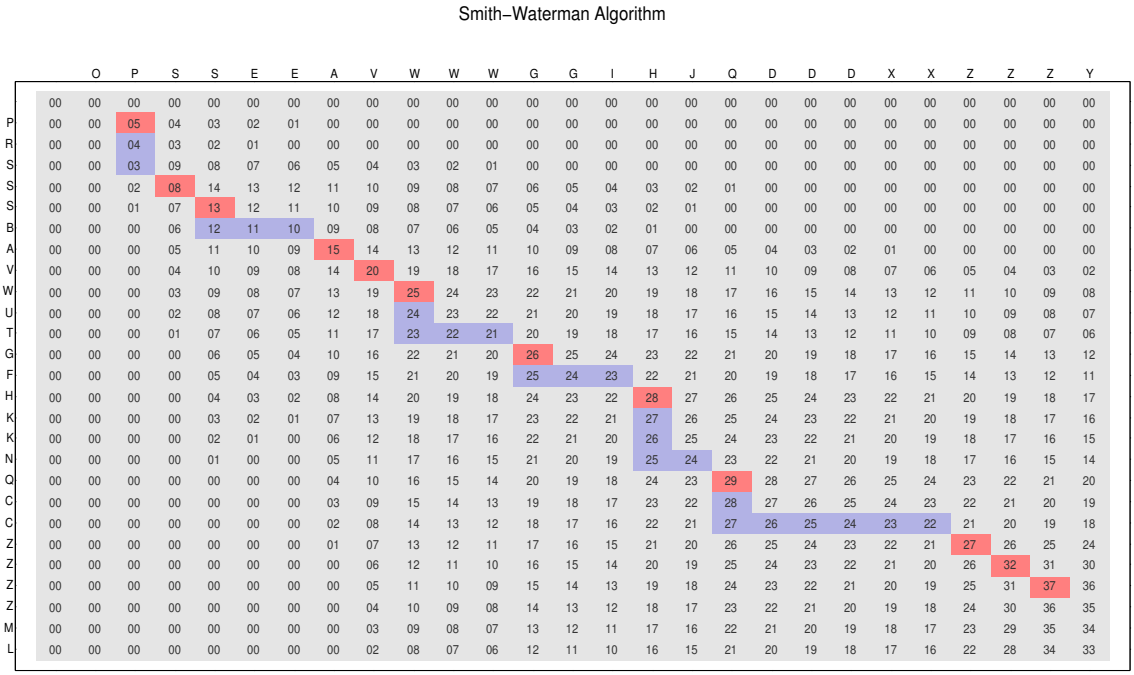


Figure 5.14: Smith-Waterman Alignment. Each symbol in the first codeword is in correspondence with one column. Similarly, symbols in the second codeword are assigned to rows in the table. The scores in each cell are computed recursively by a dynamic programming algorithm, which accounts for the cost of deletions/insertions (-1), substitutions (-2), and matches ($+5$) of symbols. A path, tracing back from the largest value in the matrix, identifies the optimal alignment of the two sequences.

Figure 5.13.

The score for an alignment ending with the symbols $w_a(i)$, $i \geq 1$ and $w_b(j)$, $j \geq 1$, is computed by the following simple recursion:

$$S_{ab}(i,j) = \max \begin{cases} S_{ab}(i-1,j-1) + C^{\mathbf{sub}}(w_a(i), w_b(j)) & \text{replace } w_a(i) \text{ with } w_b(j) \\ S_{ab}(i,j-1) + C^{\mathbf{del}} & \text{delete } w_b(j) \text{ from } w_b \\ S_{ab}(i-1,j) + C^{\mathbf{del}} & \text{delete } w_a(i) \text{ from } w_a \end{cases} \tag{5.7}$$

where $C^{\mathbf{del}}$ is the decrease in score imposed by deleting one symbol from a codeword (also interpretable as an insertion of a *blank* in the other codeword), while $C^{\mathbf{sub}}(w_a(i), w_b(j))$ is

the (possibly negative[6]) increase in score of matching the i-th symbols of w_a with the j-th symbol of w_b.

The algorithm is initialized by assuming a border of zero score to the left of the first column and top of the first row. Once the scoring matrix S_{ab} is computed, we start from its largest value and trace back through the matrix until we reach a value of zero. This process can be greatly facilitated by recording the "moves" performed in (5.7) by the algorithm at every iteration.

5.4.4.2 Agglomerative Clustering

The Smith-Waterman distance metric just introduced is an excellent tool to determine the similarity of codewords. It can handle pairs which are different in length, and allows for some shifting of descriptors during the match. One drawback, however, is the fact that it does not scale well to the alignment of multiple codewords: although its extension is trivial, the computational complexity grows exponentially in the number of sequences. The problem itself remains an open question and, once again, is of great interest in biology.

Since some clustering techniques, such as k-means or mixture of Gaussians, go through the computation of some form of average element, the alignment of multiple sequences becomes necessary. We bypass the problem by settling for an alternative clustering algorithm, known as *agglomerative clustering*, which requires nothing more than the affinity matrix.

Agglomerative clustering groups the data in a hierarchical structure called a *dendrogram*. Initially, each one of the N data points is assumed to be a cluster of its own. Then, the two most similar clusters (really, single data points, up to now) are replaced by a cluster containing them, which has the two clusters as children. This leaves $N-1$ items to be

[6]Generally, $C^{\mathbf{del}} < 0$ since we want to penalize deletions/insertions. Substitutions have score $C^{\mathbf{sub}} \geq 2C^{\mathbf{del}}$, since a deletion on each side effectively amounts to a substitution. The better the match between two symbols being exchanged, the larger the score that is assigned to it.

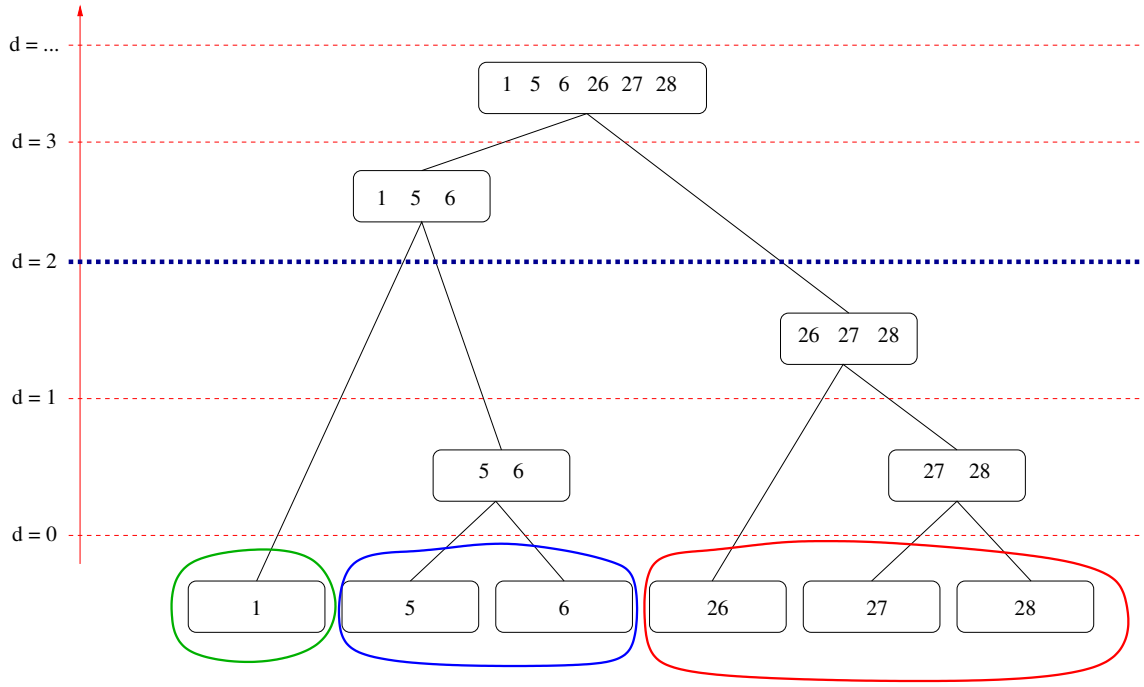


Figure 5.15: Agglomerative Clustering. Initially, each one of the six elements is a cluster. A dendrogram is constructed by recursively grouping together the two most similar clusters. Similarity is determined by the average distances of elements in one cluster to elements in the other. Finally, a threshold on maximum intra-cluster distance breaks the links at higher levels, yielding a set of clusters.

clustered. Next, the two most similar clusters are merged and replaced by their union, leaving $N - 2$ clusters. This continues until all elements are in one single cluster. When deciding which pair to merge, several criteria can be used. The *average linkage* chooses the two sets where the elements in one set have smallest average distance to the elements of the other set. When the average is replaced by the maximum or minimum, we have the *complete linkage* or *single linkage*, respectively. Figure 5.15 shows a sample hierarchy obtained by single-linkage clustering of six data points. Pairwise affinity is based on the Euclidean distance.

Once the dendrogram is constructed, clusters can be defined by choosing a threshold on the maximum intra-cluster distance, or other such criteria that eliminates links, starting from the top and descending to some level in the dendrogram. The resulting connectivity

defines what each cluster contains.

5.4.5 Experiments

In this section we show a few results obtained with the algorithm we have presented.

Although quantitative results are generally desirable, we provide only a limited, and mostly qualitative, evaluation. This is in part due to the difficulty of representing on paper what is most naturally understood through a video signal.

More importantly, we feel that the process of establishing a ground-truth decomposition into movemes is at best subjective, hence making a quantitative measure of the performance of little interest.

Finally, we observe that our focus has been directed at the discovery of movemes, which is the very first step of the activity/action hierarchy which we have hypothesized. We believe that a quantitative evaluation would be beneficial when performed on an end-to-end system, that can produce descriptions of behaviors which are at the same level of abstraction as those of a typical human being.

Representing the Dynamics :

In this experiment we apply the learning procedure of Section 5.4.1 to 17 video sequences with a total of $4,610$ frames. The input to the algorithm is a set of locations in each frame which identify the joints of the body. For each limb we learn a model of its four coordinates. Each coordinate is trained independently by means of a SLDS with five switches. The training is stopped when the likelihood increase is below a small threshold.

Figure 5.16 shows results on sequence 8. Each one of the four plots represents the probability mass of the switch for one of the coordinates within the right arm. These

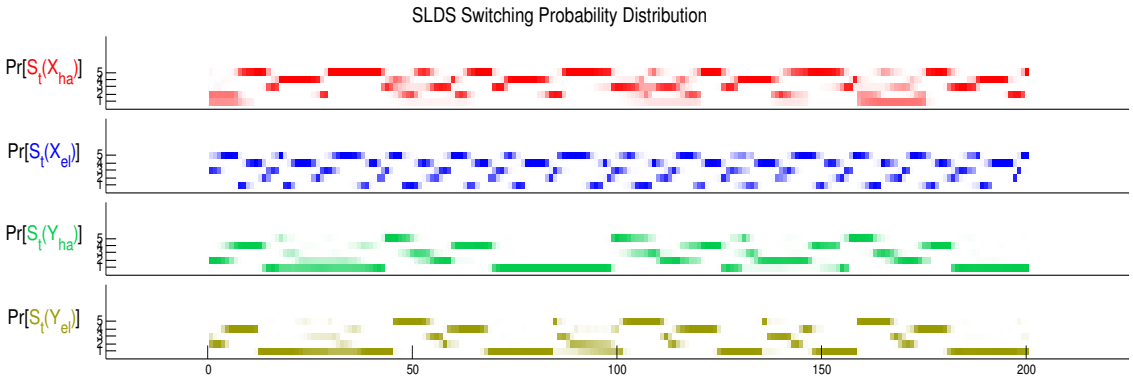


Figure 5.16: Estimating the Switches. We show experimental results on sequence 8 of 17. Each plot represents the probability mass of the switch (which takes values $1 \ldots 5$) for one of the coordinates within the right arm. All observations (including future ones) are used to compute the smooth estimate of the switching variable.

are smoothed estimates since all the observations (including future ones) are used. A glance at the plots (qualitatively) shows that the repetitive nature of the motion is captured by the switches alternating over time.

Moveme Representation of Video Sequence :

In this experiment we show a decomposition in movemes of a video sequence. As before, we train our model for the right arm on a total of $4,610$ frames, obtaining a codeword-based representation of the data. The codewords are clustered according to the procedure detailed in Section 5.4.4. We set the number of clusters to 15. This results in 10 visually distinct (to a human) motions. The reason for the lower number is that some of the motions are represented by multiple clusters. By observing the fragment of video contributing to each cluster, we manually assign a textual label to each cluster with the purpose of describing its content.

The vertical axis of Figure 5.17 lists in colors the subset of movemes that appeared in sequence 8. The top half reports the raw trajectories of the four coordinates in time. At the bottom, the height of each bar corresponds to a different (to a human) moveme.

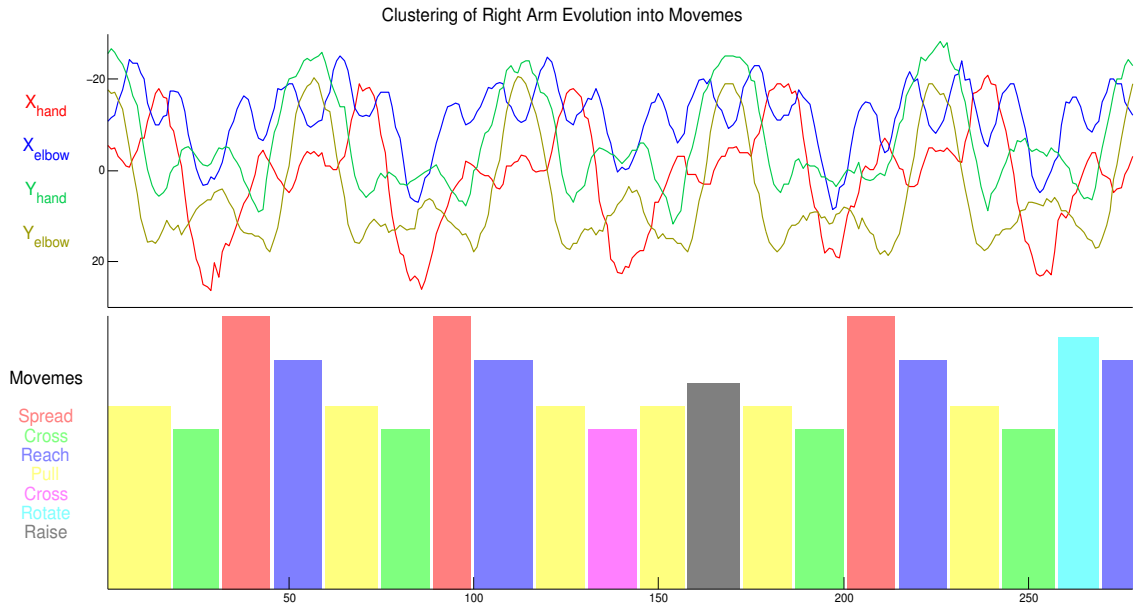


Figure 5.17: Decomposition into Movemes. We show a moveme-based representation of the motions in sequence 8. At the top we report the trajectories of the right arm's four coordinates. The bottom plot shows the movemes over time. Height indicates the identity of the moveme, as perceived by a human. Names for the movemes are provided on the vertical axis. The 10th bar, which represent an "arm crossing", is (to a human) visually similar to the 2nd, 6th, and 14th (they have the same height). However, its color is different since the model returned a different moveme/cluster for its representation.

Notice how the 2nd, 6th, 10th, and 14th bars have identical height, indicating that a human would consider all four motions as a "crossing of the arm in front of the chest". Yet, bar number 10 carries a different color, since our algorithm produced multiple movemes (two in this case) representing the same motion.

5.5 Discussion

In the previous sections, we have presented our approach to the discovery of movemes, which is based on a three-step process. At first, the sequence is represented by a set of switching linear dynamical systems, which encode the limb's dynamics with a small descriptor. The sequence of descriptors is then heuristically segmented into codewords by means of an energy

that is derived from the velocity of the parts in the limb. Finally, the codewords are grouped together based on their similarity and the resulting clusters, which we call movemes, identify the stereotypical motions appearing in the sequence.

Although the results of this process are visually appealing, we see an interesting opportunity for improvement. More specifically, we would like to rid ourselves of both the heuristic-based segmentation and the sub-optimal definition of movemes, which are consequences of handling the identification and grouping of codewords as two independent problems. Borrowing from the fields of data mining and text analysis [GPS99], we present a few ideas on how to approach the problem of simultaneous segmentation and clustering.

For many languages, it is easy to identify words in a written text, since they are separated by spaces. This, however, is not always the case. In Chinese, for example, a sentence is a sequence of characters delimited by periods or commas. In order to perform any type of analysis on the text, one needs to determine a segmentation of the text into words. This is particularly challenging, since the length of each word can vary from as little as one to as many as four or five characters. Furthermore, characters that alone define a word, and have a meaning of their own, can also belong to longer words, making their interpretation different depending on context.

Most algorithms that deal with the automatic analysis of written Chinese require that a segmentation of the text into words be provided. Other approaches use a dictionary to produce a segmentation of the text into words, and then proceed with the analysis. In our situation, however, we must not rely on such information being available, since a dictionary is exactly what we are trying to build from the text.

When approaching this problem, one simplifying assumption is to let the words occur independently of each other. In [GPS99] it is observed that the task of segmenting the

text can then be easily completed, if a probability of occurrence for each word is given. Alternatively, given the segmented text, it is easy to compute the probability of occurrence for each word, by simply counting the number of times that word appears. Once again, this is the dilemma we have been seeing all along in this thesis, and by now we know that EM is most likely the answer. The probabilities of each word can be taken as the unknown parameters in the model, while the locations of the codewords' boundaries are the hidden variables. Iterating between the two steps of EM yields both the probabilities of the words, and the most likely segmentation. In [GPS99] it is shown how to efficiently perform the two steps with dynamic programming.

Although this idea seems to take us in the right direction, we have skipped over a number of details. The most delicate issue is with the choice of probability model for the moveme. While with written Chinese, multiple repetitions of the same words appear as the same concatenation of identical characters, this is not at all the case with the descriptors in our video sequences. In fact, the type and number of descriptors, contained in a given moveme, can vary slightly from one codeword instance to another, leaving us with the task of somehow equating and combining different codewords into one probabilistic model for the moveme.

One approach could be to represent each moveme with a profile hidden Markov model (P-HMM). Here the parameters become the probability of occurrence of the moveme, as well as the stochastic representation of its codewords by the P-HMM.

Given a model, we can establish the probability of any codeword being an instance of that moveme, by computing the likelihood of the codeword in the P-HMM. This allows us to assign that codeword to its most likely moveme. Of course "soft assignments" are also possible, and perhaps preferrable. Once each codeword is mapped to a moveme, multiplying

together the probabilities of occurrence of the movemes gives the likelihood of the segmentation. Since the likelihood can efficiently be computed for every possible segmentation, this concludes the E-step of EM.

In the M-step, we need to update the probability of occurrence of each moveme, as well as the parameters of their P-HMM representation. The former can be done efficiently as detailed in [GPS99]. The latter, however, requires us to align the exemplar codewords to each other, so that a P-HMM can be re-estimated. As we mentioned in previous sections, the alignment of multiple sequences remains an open question. Nevertheless, this can be accomplished approximately by means of some heuristics.

Although many details are missing to reach a full end-to-end solution for the simultaneous segmentation and grouping of codewords into movemes, we conclude this section feeling that the process we have outlined is a very promising approach worth exploring. We leave that for future investigation.

Chapter 6

Conclusions and Future Work

In this thesis, we have presented a probabilistic approach to the problem of detection and labeling of human motion, and its representation by means of atomical motions, or movemes.

Under the assumption that the human body can be decomposed into a set of parts, we have used the formalism of graphical models to represent the dependencies among them. In the first part of our work, we have explored the benefits of allowing different degrees of interactions among the parts, which we represent by means of (loopy) graphical structures. A second contribution is the introduction in the model of global variables. These properties, which could include aspects such as size, viewpoint, or center of the person, act as a cohesive element among the parts, increasing robustness to occlusion and preventing degenerate situations from occurring. With the enhanced hybrid model, we have also presented a couple of ways of doing inference, which combine the efficiency of the junction tree algorithm with the power afforded by Monte Carlo Markov chain sampling methods.

In the second part of our work, we have tackled the problem of representing actions and activities in video sequences. We hypothesize a hierarchical representation of motion, based on the temporal extension and semantic interpretation of events. At the bottom layer of the hierarchy are the *movemes*: atomical motions which act as building blocks in the representation of actions and activities.

We have presented two approaches to the unsupervised discovery of movemes from video sequences. Our first attempt relies on a global variables to identify the moveme being performed at each instant in time. The EM algorithm is used to infer the membership of each frame in the sequence to one of the movemes. This produces a representation of the sequence as a probabilistic concatenation of movemes.

Our second attempt to the unsupervised discovery of movemes is driven by the realization that dynamic information is an integral part of the motion and is, at times, indispensible in discriminating between motions that traverse a similar set of poses, but with very different semantics. In order to transduce the dynamical information in a compact representation we have used switching linear dynamical systems (SLDS). After introducing the general theory of an SLDS, we have shown how to learn the parameters from data, and how to do inference in such models. Following the segmentation and transduction of each limb's dynamics into codewords, we have shown how a simple sequence-alignment algorithm, which we borrow from the field of biology, can be used to robustly compare codewords of different lengths, even in the presence of small amount of internal shifting and mutations due to noise. Finally, we have described a standard aggregation procedure that groups similar codewords into clusters, which we take as the definition of movemes.

Although we feel our work provides a good example of what type of information can be automatically extracted from the analysis of video, in Sections 2.4 and 5.5 we have mentioned a few interesting extensions and direction of research, which we leave for future exploration.

Bibliography

[AC99] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[AM00] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[AT04] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Computer Vision and Pattern Recognition*, pages 882–888, 2004.

[Bak89] H. H. Baker. Building surfaces of evolution: The weaving wall. *International Journal of Computer Vision*, 3(1):51–72, May 1989.

[Bar78] Y. Bar-Shalom. Tracking methods in a multitarget environment. *IEEE Transaction on Automatic Control*, 23(4):618–626, Aug 1978.

[BCMS01] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *Computer Vision and Pattern Recognition*, volume 02, page 52, 2001.

[BCS07] A. Bissacco, A. Chiuso, and S. Soatto. Classification and recognition of dynamical models: The role of phase, independent components, kernels and optimal transport. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1958–1972, 2007.

[BD01] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.

[BGS+05] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *International Conference of Computer Vision*, pages 1395–1402, 2005.

[BI96] A. Blake and M. Isard. The condensation algorithm—conditional density propagation and applications to visual tracking. In Michael Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Neural Information Processing Systems*, pages 361–367. MIT Press, 1996.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[Bis07] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[BJ02] F. R. Bach and M. I. Jordan. Learning graphical models with Mercer kernels. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Neural Information Processing Systems*, pages 1009–1016. MIT Press, 2002.

[BK98] X. Boyen and D. Koller. Approximate learning of dynamic models. In *Neural Information Processing Systems*, pages 396–402. MIT Press, 1998.

[BL93] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques and Software*. Artech House, 1993.

[Bob97] A. F. Bobick. Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of Royal Society of London*, 352(1358):1257–1265, Aug 1997.

[BOP97] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[Bre97] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Computer Vision and Pattern Recognition*, pages 568–574. IEEE Computer Society, 1997.

[BW97] A. F. Bobick and A. D. Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, 1997.

[BYJ97] M. J. Black, Y. Yacoob, and S. X. Ju. Recognizing human motion using parameterized models of optical flow. In M. Shah and R. Jain, editors, *Motion-based Recognition*, pages 245–269. Kluwer Academic Publishing, 1997.

[CD00] R. Cutler and L. S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, August 2000.

[Chi96] D. M. Chickering. Learning Bayesian networks is np-complete. In D. Fisher and H. J. Lenz, editors, *Learning from data: Artificial Intelligence and Statistics*, chapter 12, pages 121–130. Springer-Verlag, 1996.

[CK77] J. E. Cutting and L. T. Kozlowski. Recognizing friends by their walk: Gait perception without familiarity cues. *Bulletin Psychonometric Society*, (9):353–356, 1977.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[DMP02] D. Del Vecchio, R. M. Murray, and P. Perona. Primitives for human motion: A dynamical approach. In *IFAC World Congress*, Barcelona, Spain, 2002.

[DMP03a] D. Del Vecchio, R. M. Murray, and P. Perona. Classification of human motion into dynamics-based primitives with application to drawing tasks. In *European Control Conference*, Cambridge, UK, 2003.

[DMP03b] D. Del Vecchio, R. M. Murray, and P. Perona. Decomposition of human motion into dynamics-based primitives with application to drawing tasks. *Automatica*, (39):2085–2098, July 2003.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, pages 886–893. IEEE Computer Society, 2005.

[DTLM96] W. Dittrich, T. Troscianko, S. Lea, and D. Morgan. Perception of emotion from dynamic point-light displays represented in dance. *Perception*, (25):727–738, 1996.

[DTTB07] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *Computer Vision and Pattern Recognition*, June 2007.

[EBMM03] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[FAI+05] D. A. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan. *Computational Studies of Human Motion: Part I, Tracking and Motion Synthesis*, volume 1, pages 77–254. Now Publishers Inc., 2005.

[FE73] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, 22(1):67–92, January 1973.

[FH00] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Computer Vision and Pattern Recognition*, pages 66–73, 2000.

[For86] W. Forstner. A feature-based correspondence algorithm for image matching. *International Archives of Photogrammetry and Remote Sensing*, 3(26):150–166, 1986.

[FP02] X. Feng and P. Perona. Human action recognition by sequence of movelet codewords. In *First International Symposium on 3D Data Processing, Visualization and Transmission*, pages 717–723. IEEE Computer Society, 2002.

[Gav99] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[Gav00] D. Gavrila. Pedestrian detection from a moving vehicle. In *European Conference of Computer Vision*, pages 37–49, 2000.

[GC03] P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Journal of Machine Learning Research*, 50(1-2):127–158, 2003.

[GH96] Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, Deptartment of Computer Science, Univeristy of Toronto, 1996.

[GH98] Z. Ghahramani and G. E. Hinton. Switching state-space models. Technical Report CRG-TR-96-3, Deptartment of Computer Science, University of Toronto, 1998.

[GH00] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[Gha98] Z. Ghahramani. Learning dynamic bayesian networks. In *Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks, "E.R. Caianiello"—Tutorial Lectures*, pages 168–197. Springer-Verlag, 1998.

[GPS99] X. Ge, W. Pratt, and P. Smyth. Discovering chinese words from unsegmented text. In *Poster abstract at the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 271–272. ACM, 1999.

[HS88] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[IF01] S. Ioffe and D. A. Forsyth. Human tracking with mixtures of trees. In *International Conference of Computer Vision*, pages 690–695, 2001.

[IF07] N. Ikizler and D. A. Forsyth. Searching video for complex activities with finite state models. In *Computer Vision and Pattern Recognition*. IEEE Computer Society, 2007.

[Joh73] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[Jor] M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. (In preparation.).

[Jor99] M. I. Jordan. *Learning in Graphical Models*. MIT Press, 1999.

[Kim94] C.-J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60(1-2):1–22, 1994.

[Lau96] S. L. Lauritzen. *Graphical Models*. Oxford University Press, USA, July 1996.

[LB94] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–294, 1994.

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

[LSS05] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition*, pages 878–885, 2005.

[MFM04] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[MM94] G. Mather and L. Murdoch. Gender discrimination in biological motion displays based on dynamic cues. In *Royal Society of London*, number 259 in B, pages 273–279, 1994.

[MM02] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *European Conference of Computer Vision*, volume 2352, pages 666–680. Springer, 2002.

[MREM04] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *Computer Vision and Pattern Recognition*, pages 326–333, 2004.

[MSSS04] T. Mori, Y. Segawa, M. Shimosaka, and T. Sato. Hierarchical recognition of daily human actions based on continuous hidden Markov models. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 779–784, 2004.

[MSZ04] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference of Computer Vision*, volume 3021, pages 69–82. Springer, 2004.

[Mur98] K. P. Murphy. Switching Kalman filters. Technical Report 98-10, Compaq Research Laboratory, Aug 1998.

[Mur02] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, July 2002.

[MWJ99] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In Kathryn B. Laskey and Henri

Prade, editors, *Uncertainty in Artificial Intelligence*, pages 467–475. Morgan Kaufmann, 1999.

[NA94] S. A. Niyogi and E. H. Adelson. Analyzing and recognizing walking figures in xyt. In *Computer Vision and Pattern Recognition*, pages 469–474, 1994.

[NBIR00] B. North, A. Blake, M. Isard, and J. Rittscher. Learning and classification of complex dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1016–1034, 2000.

[ORBD05] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inference in parametric switching linear dynamical systems. In *International Conference of Computer Vision*, pages II: 1161–1168, 2005.

[Pas03] M. A. Paskin. Sample propagation. In *Neural Information Processing Systems*. MIT Press, 2003.

[Pea88] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[PP00] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[PR00] V. Pavlovic and J. M. Rehg. Impact of dynamic model learning on classification of human motion. In *Computer Vision and Pattern Recognition*, pages I:788–795, 2000.

[RFZ05] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Computer Vision and Pattern Recognition*, pages 271–278. IEEE Computer Society, 2005.

[Roh97] K. Rohr. *Human movement analysis based on explicit motion models*, chapter 8, pages 171–198. Kluwer Academic Publishers, 1997.

[Sch78] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[SGP00] Y. Song, L. Goncalves, and P. Perona. Monocular perception of biological motion—clutter and partial occlusion. In David Vernon, editor, *European Conference of Computer Vision*, volume 1843 of *Lecture Notes in Computer Science*, pages 719–733. Springer, 2000.

[SGP01a] Y. Song, L. Goncalves, and P. Perona. Learning probabilistic structure for human motion detection. In *Computer Vision and Pattern Recognition*, pages 771–777. IEEE Computer Society, 2001.

[SGP01b] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion models. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Neural Information Processing Systems*, pages 1287–1294. MIT Press, 2001.

[SS82] R. Shumway and D. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[SS91] R. Shumway and D. Stoffer. Dynamic linear models with switching. *Journal of the American Statistical Association*, 86(415):763–769, Sept 1991.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. In *Journal of Molecular Biology*, volume 147(1), pages 195–197, 1981.

[TK91] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[VJ01] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[VJS05] P. A. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[WB95] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill, 1995.

[Wel00] M. Welling. Learning System: Caltech CS 156B Class Notes on EM, Spring 2000.

[WF01] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

[WWP00] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In David Vernon, editor, *European Conference of Computer Vision*, volume 1842 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2000.

[YFW00] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Neural Information Processing Systems*, pages 689–695. MIT Press, 2000.

[YFW05] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[YOI92] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[YXC97] J. Yang, Y. Xu, and C. S. Chen. Human action learning via hidden Markov model. *IEEE Transaction on System, Man, and Cybernetics*, 27(1):34–44, 1997.

[ZT00] L. Zhao and C. Thorpe. Stereo and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3):148 –154, September 2000.

[ZT01] J. M. Zacks and B. Tversky. Event structure in perception and conception. *Psychological Bulletin*, 127:3–21, 2001.