LEARNING STRUCTURED MODELS FOR HUMAN ACTIONS AND POSES

by

Yang Wang

B.Eng., Harbin Institute of Technology, China, 2002

M.Sc., University of Alberta, Canada, 2004

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2009

APPROVAL

Name:	Yang Wang
Degree:	Doctor of Philosophy
Title of Thesis:	Learning Structured Models for Human Actions and Poses

Examining Committee: Dr. Brian Funt
Chair

____
Dr. Greg Mori, Senior Supervisor

____
Dr. Anoop Sarkar, Supervisor

____
Dr. Mark S. Drew, SFU Examiner

____
Dr. David Forsyth, External Examiner
Professor of Computer Science,
University of Illinois at Urbana-Champaign

Date Approved: Aug 31, 2009

Abstract

A grand challenge of computer vision is to enable machines to “see people”. A solution to this challenge will enable numerous applications in various fields, e.g., security, surveillance, entertainment, human computer interaction, bio-mechanics, etc.

This dissertation focuses on two problems in the general area of “looking at people”, *human pose estimation* and *human action recognition.* The first problem is to identify the body parts of a person from a still image. The second problem is to recognize the actions of a person from a video sequence. We formulate the solutions to these problems as learning *structured models.* In particular, we propose models and algorithms to address the following structures: (1) human pose estimation as structured output problem. We propose a boosted multiple tree model for modeling the spatial and occlusion constraints between human body parts; (2) temporal structure in human action recognition. We present two models based on the “bag-of-words” representation to capture the temporal structures of video sequences; (3) human action recognition as classification with hidden structures. We develop a model based on the hidden conditional random field to recognize human actions. We also propose a max-margin learning method for training the model. The learning method is general enough to be applied in many other applications in computer vision, even other areas in computer science.

Keywords: computer vision; machine learning; human pose estimation; human action recognition

Acknowledgments

First and foremost I would like to thank my PhD advisor, Professor Greg Mori, for being the best advisor anyone could ask for. Throughout the past few years, Greg has helped my academic pursuit in every possible aspect. I would not have come this far without the guidance, support, and plenty of research freedom he has given me. Having Greg as my PhD advisor is no doubt the best decision I have ever made in my PhD career, and I am proud to be his first PhD graduate. If one day I have the chance to advise students of my own and I can have half of Greg's patience and effort, I will consider it a success.

Thanks to Professors Anoop Sarkar and Mark Drew for serving on my thesis committee and for their constructive comments on my research. Thanks to Professor Ze-Nian Li for many advices and comments in the past few years. Thanks to Professor Brian Funt for chairing my thesis defense.

Thanks to Professor David Forsyth for serving as my external examiner. David's pioneering work in many areas of computer vision has greatly influenced my own research. It is a great honor to have him on my thesis committee.

Shaojun Wang and GholamReza Haffari are my main collaborators in machine learning in the last few years. I have greatly enjoyed our fruitful collaborations and many three-way email discussions. Although not reported in this thesis, those collaborations have added an extra dimension to my research and helped me to be a more well-rounded researcher.

Thanks to all the members of SFU Vision and Media Lab for creating a cooperative and friendly environment to work. I have also enjoyed collaborating with many lab members on various projects, including (in alphabetical order) Hao Jiang, Tian Lan, Mohammad Norouzi, Mani Ranjbar, Payam Sabzmeydani, Weilong Yang. Especially thanks to Hao Jiang for his help when I first started in computer vision, and for many advices and suggestions over the years. Also thanks to all my friends for making my stay at SFU a wonderful experience.

Thanks to Drs. Andrew Blake and Shahram Izadi for hiring me as a research intern at Microsoft Research Cambridge. Spending the summer at Cambridge was a wonderful experience. Also thanks to Dr. Ollie Williams for his help during my internship.

Last but not the least, thanks to my family for many years of support and sacrifice. I dedicate this disseration to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The goal of computer vision is to enable computers to see, i.e., to analyze and understand images and videos. In this dissertation, we are particularly interested in the problem of enabling computers to see *people*. Given an image (or a video) containing people, we would like to have a computer algorithm that can automatically localize persons in the image, figure out their body poses, and recognize their activities. In computer vision literature, these problems related to humans are broadly called "looking at people". There are many reasons one would like to study people. The most important one is that there are many real-world applications in computer vision that involve people. Table 1.1 lists some example applications that can potentially benefit from computer algorithms that can "see and understand people" from image/video data. Another reason is that people are typically the dominant objects in the visual data (images or videos) we encounter everyday, e.g., movies, TV, news images. We believe understanding images and videos of people is a very crucial part of understanding the visual world.

Enabling computers to see people is an extremely challenging problem. The general area of "looking at people" in computer vision comprises many aspects. Most of the research efforts in the computer vision community have focused on the following subareas: *human detection*, *human tracking* (both blob tracking and kinematic tracking), *human pose estimation*, *human action recognition*. Although they have been studied as separate research problems, these subareas are closely interleaved – the solution of one problem can typically provide useful information for solving another problem, please refer to some discussions in Section 1.1. In this dissertation, we limit ourselves to *human pose estimation* and *human action recognition*.

Application domain	Potential use
security, surveillance	automatic monitoring, abnormality detection
human computer interaction	human gestures as input devices
sports, entertainment	markerless motion capture and sports analysis
Web application	movie/video retrieval

Table 1.1: Examples of application domain that can benefit from computer vision systems that can understand people.

1.1 Background

Human Pose Estimation: *Human pose estimation* is a problem of identifying the body parts (or joints) of a person from a still image, see Figure 1.1. Human pose estimation is closely related to other subareas of "looking at people". When human poses are estimated over time (e.g., in a video sequence), the term *(kinematic) human tracking* is typically used. When the whole body of a person is perceived as a single object, the problem of *human pose estimation* becomes *human detection*. The output of a successful human pose estimation system provides valuable information to solve other subareas of "looking at people". For example, if we can do human pose estimation reliably, *human tracking* will become easy–we can simply run the pose estimation algorithm on each frame individually. Human pose estimation also provides an automatic way to initialize a tracker. This line of work has been explored in [81, 82]. The problem of *human detection* can also greatly benefit from human pose estimation. For example, the work in [106] demonstrates that even a rough estimate of body part positions can help improving state-of-the-art human detection systems. The output from a human pose estimation system can also be used for *human action recognition*, e.g., [79, 41]. In this dissertation, we mainly focus on human pose estimation in 2D, although in the literature, there is also work on human pose estimation in 3D [2], and work on obtaining the segmentation mask associated with human figures [68].

Human pose estimation is arguably one of the most difficult problems in computer vision. The difficulties are manifold. The first and probably the most important difficulty is the huge amount of intra-class variation. Intra-class variation is a challenge that many recognition problems in computer vision have to deal with. In human pose estimation, this difficulty is amplified by the fact that humans are articulated objects, and can bend and contort their bodies into a wide variety of poses. See Figure 1.2 for some examples. Due to this difficulty, it is extremely difficult to build a model that can generalize to all

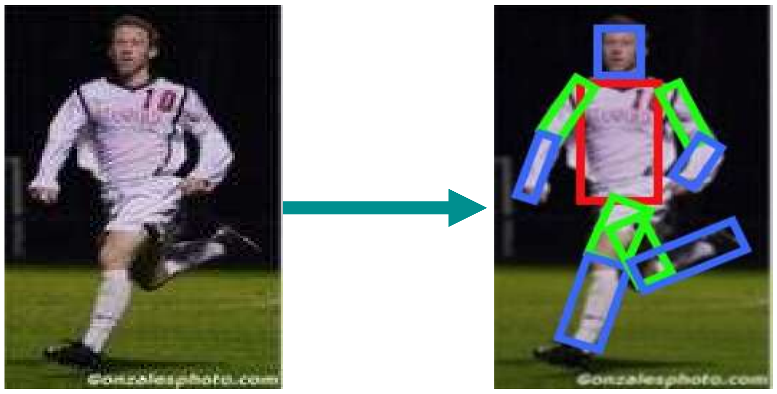


Figure 1.1: The goal of human pose estimation is to identify the body parts from an input still image.

the possible human poses. Secondly, the parts which make up a human figure are varied in appearance (due to clothing), which make it difficult to reliably detect them. Thirdly, parts often have small support in the image or are occluded. Moreover, an image typically contains a lot of limb-like background clutters – things (e.g., buildings, trees) that look like body parts.

Human Action Recognition: The goal of human action recognition is to classify a video clip (or each individual frame of the video) into one of pre-defined categories according to the action performed by the person in the video, see Figure 1.3. Before we proceed, we first need to clarify the terminology we are using. In this dissertation, we use the term "action" to describe simple, atomic movement (e.g., running, walking, hand-waving, etc). We use the term "activity" to refer a series of atomic actions for achieving a particular goal. For example, the "activity" of withdrawing cash from an ATM machine involves several "actions"– walking to the ATM, inserting the bank card, entering the PIN number, and taking the cash. An even more complicated problem is to recognize the "group activity" involving more than one person. These definitions are not unique. In the literature, people sometimes use these terms interchangeably. In this dissertation, we use the term "human action recognition" to refer to the problem of classifying simple, atomic "actions". Recognizing atomic actions is one important part of the overall hierarchy in understanding human activities.

Human action recognition is also closely related to other subareas of "looking at people". E.g., human pose estimation is a very hard problem, but estimating the human pose of a person performing a particular action (e.g., walking) is relatively easy, since we can build

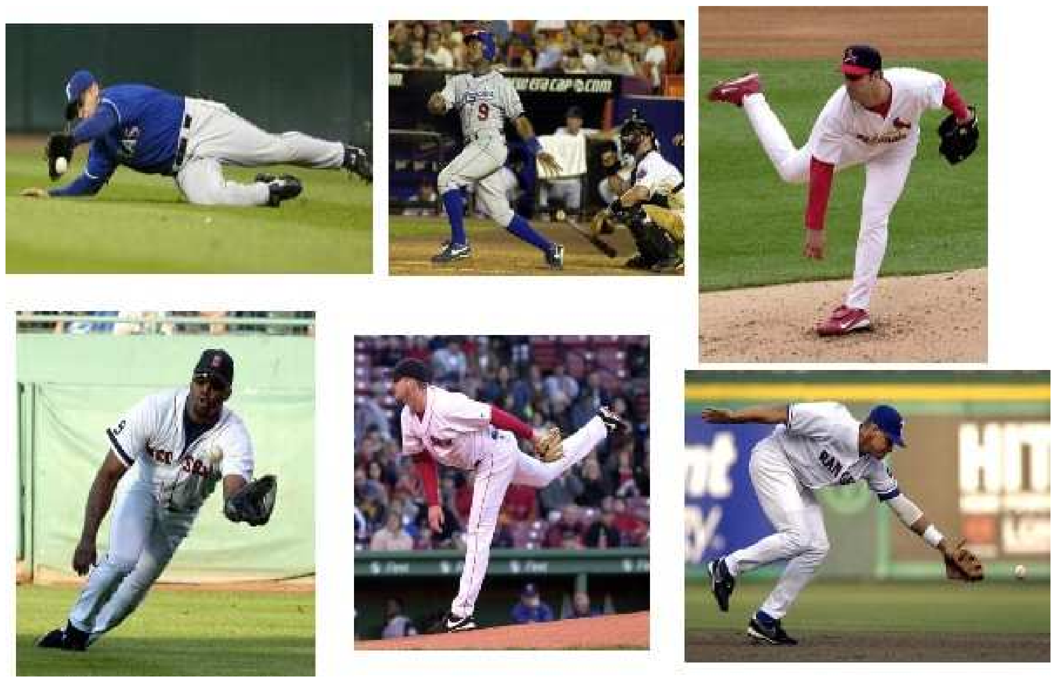


Figure 1.2: Examples demonstrating the challenges of the human pose estimation problem.

a pose estimation system specifically tuned to the walking action [51]. If we have access to a reliable human action recognition system, we can use it as the input to a human pose estimation algorithm. Similarly, if we know the action being performed by a person, the problem of human tracking also becomes easy, since we can build an "action-specific" tracker that exploit the temporal information specifically designed for that action.

Recognizing human actions is also a very challenging problem. The first challenge comes from the intra-class variations. Consider the "walking" action as an example. People can walk with different speeds, styles, and strides. They can wear different clothes. The lighting and environment might be different. A successful human action recognition system needs to be able to differentiate between different action classes without being affected by all the nuances. In many real-world scenarios, we also need to cope with camera motions, background clutters, low resolution images, occlusions, etc.

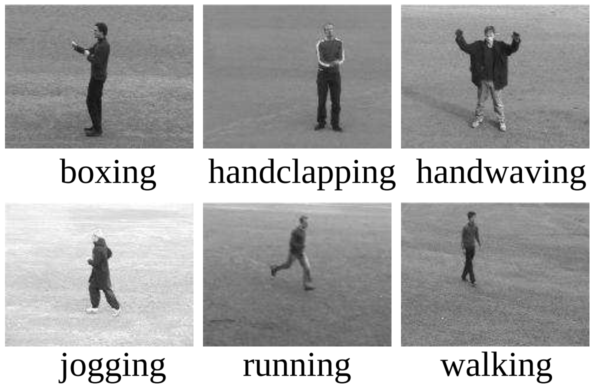


Figure 1.3: The goal of human action recognition is to identify the type of action the person is performing from the video data.

1.2 Contributions

The field of computer vision has been deeply influenced by the rapid growth of the machine learning techniques in recent years. In fact, many successful stories (e.g., face detection, object recognition) in computer vision are inevitably associated with some techniques (e.g., boosting, SVM) in the machine learning literature.

Despite of the great success, we see a gap between learning and vision. Typical machine learning techniques aim to learn a mapping function $f : X \rightarrow Y$, which maps the input X to the output space Y. The input X is usually a high dimensional vector, i.e., $X \in \mathbb{R}^n$. The output Y can be a set of discrete labels in the case of classification, or a continuous number in the case of regression.

Unfortunately, this elegant mathematical assumption of most machine learning techniques is only a simplified version of the real world. Our visual world is clearly not an n-dimensional vector. A lot of the vision tasks involve more than just a single number (discrete or continuous) as the output. The inputs and outputs of many vision problems usually interact in a complex way that involve many latent structures. Many works in computer vision ignore such meaningful structures and simply "stack up" the image pixels (or other feature representation) and use some standard machine learning techniques as a black-box tool. This "black-box" view of machine learning in computer vision has been popular in the past few years and has achieve a significant amount of successes – a representative one being the whole body of work on "bag-of-words" approaches in visual recognition. However, we believe there is a limit on how far we can go with this line of work.

The major contribution of this dissertation is on learning *structured models* for solving vision problems. Instead of feeding our image data into a "magic" black-box machine learning toolbox, we choose to explicitly model the kind of rich structures inherent in the problems we are solving. Although we focus on human pose estimation and human action recognition, most of the techniques developed in this context go beyond the scope of dissertation and can be broadly applied in many other computer vision problems, even other areas in computer science. In particular, we will address three different structures: structured outputs, structured latent variables, and temporal structures. We provide a series of models and algorithms that explicitly exploit these structures for solving our problems.

1.3 Outline

The rest of this dissertation is organized as follows:

Chapter 2 provides an overview of previous work in human pose estimation and human action recognition that is most relevant to this dissertation.

Chapter 3 focuses on human pose estimation from still images. We formulate the human pose estimation problem as a learning problem with structured outputs. Many existing approaches use tree-structured probabilistic models to solve this problem. This is mainly because there are efficient learning and inference algorithms for these types of models. However, one limitation of tree-structured models is that they cannot capture certain constraints between body parts, other than the kinematic constraints. In this chapter, we develop a boosted multiple tree framework for solving this problem. Instead of using one tree-structured model, we combine multiple tree models learned in a discriminative way by boosting. Our final model can alleviate the limitation of previous tree-structured models used in this area. This work is published as [111, 112].

Chapter 4 focuses on human action recognition using "bag-of-words" representation and topic models. There has been a lot of work on visual recognition using topic models in the computer vision literature. The limitation of previous work is that their models are learned in an unsupervised fashion, and the recognition is achieved using some heuristics. In this chapter, we propose two models for human action recognition using *supervised* topic models. Compared with previous work, our models can naturally exploit the class labels during training. This work is published as [116, 115].

Chapter 5 proposes a part-based model for frame-based action recognition. Part-based models are popular in object recognition. In this work, we demonstrate that they can also be very useful for action recognition. Our model is based on the hidden conditional random field for object recognition. As far as we know, this is the first time that part-based representation is successfully applied for action recognition. This work is published as [113].

Chapter 6 builds upon the work in Chapter 5 and introduces a max-margin learning method for training the part-based model for action recognition. The learning method proposed in this chapter is general enough to be used in many other applications in computer vision and other areas in computer science. This work is published as [114].

Chapter 2

Previous Work

In this chapter, we give a general overview of previous work on human pose estimation (Section 2.1) and human action recognition (Section 2.2).

2.1 Human Pose Estimation

Some approaches of pose estimation explicitly specify a model defining what each body part looks like, and how the body parts are related. Here we call them "model-based approaches". They usually consider the human body as an assembly of parts, connected in some fashion. These approaches specify a model that relates different body parts by various constraints. The typical constraints used are kinematic constraints between connected parts, such as torso-upper half-limb connection, or upper-lower half-limb connection. But other constraints can be used as well, such as appearance constraints, scale constraints, occlusion reasoning, etc. In this framework, these approaches try to estimate the human pose by finding the optimal configuration under the model.

Model-based pose estimation can proceed in either top-down or bottom-up fashion. In a top-down approach, the model as a whole is evaluated on all the possible configurations in an image. In the bottom-up approach, candidate parts are first detected by some lower-level part detectors, then assembled together to form a valid body configuration. Since the solution space of human poses is extremely huge, different approaches have been proposed to explore this huge space. In this chapter, we try to break down the literature of model-based pose estimation into three categories: global optimization approaches (Sec. 2.1.1), bottom-up approaches (Sec. 2.1.2), and sampling-based approaches (Sec. 2.1.3). We should note

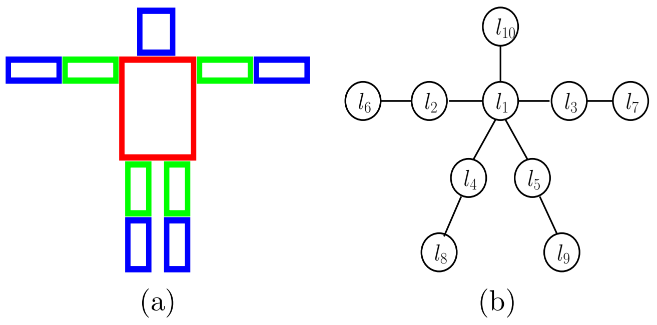


Figure 2.1: Representation of a human body: (a) human body represented as a 10-part model; (b) corresponding kinematic tree structured model.

that the distinctions between these categories are somewhat arbitrary, and certainly blur at some point. For example, many bottom-up approaches also use some sampling techniques to do the learning or inference.

2.1.1 Global optimization

The first class of approaches are characterized by the fact that they set up the problem of finding human pose as an optimization problem, then try to find the global optimum. They carefully design the objective function and the structure of the optimization problem, so it is easily solvable (or at least can be reasonably approximated).

Pictorial structures

The general framework of pictorial structures (PS) dates back to Fischler & Elschlager [33]. Felzenszwalb & Huttenlocher [30] extend this framework for handling the special case of 2D human pose estimation. In the case of tracking people, it is sometimes called a "cardboard people" model [46]. In the following, we summarize the main idea of pictorial structures presented in Felzenszwalb & Huttenlocher [30].

A pictorial structure model for an object is defined by a collection of parts with connections between certain pairs of parts. It can be naturally represented by a graph $G = (V, E)$, where the vertex $v_i \in V$ corresponds to the i-th part, and an edge $(v_i, v_j) \in E$ corresponds to the connection between parts v_i and v_j.

Under the assumption of a tree-structured graphical model, e.g., the kinematic tree (Fig. 2.1), the optimal pose L^* can be found by solving the following energy minimization problem:

$$L^* = \arg\min \left(\sum_{v_i \in V} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right) \tag{2.1}$$

Here $m_i(l_i)$ measures the degree of mismatch of placing part v_i at location l_i in the image I. $d_{ij}(l_i, l_j)$ measures the degree of deformation of the model when placing part v_i at location l_i and part v_j at location l_j, which captures the kinematic constraints.

The energy minimization formulation in Eq. 2.1 can be alternatively viewed in a statistical formulation. Let θ be the set of parameters in the model, I be an image, and $L = \{l_i\}_{i\in V}$ be a configuration of all the parts of the object. The distribution $P(L|\theta)$ measures the prior probability of the possible locations of L without seeing an image. The distribution $P(I|L,\theta)$ defines the image likelihood, i.e., the probability of seeing the image I given the fact that the locations of object parts are captured by L. Finally, we are interested in the probability $P(L|I,\theta)$, which captures the probability of object configuration L given the model parameters θ and the image observation I. $P(L|I,\theta)$ can be calculated using Bayes' rule:

$$P(L|I,\theta) \propto P(I|L,\theta)P(L|\theta) \tag{2.2}$$

The model is parameterized by $\theta = (u, E, c)$, where $u = \{u_i\}_{i\in V}$ are the appearance parameters for each part v_i. E is the set of edges indicating the connection of parts. $c = \{c_{ij}|(v_i, v_j) \in E\}$ are the parameters on the connections.

In order to make the learning and inference in the model easier, and to avoid the problem of overfitting, the appearance model $P(I|L,\theta)$ is usually assumed to take the following factorized approximation:

$$P(I|L,\theta) = P(I|L,u) \propto \prod_{i=1}^{n} P(I|l_i, u_i) \tag{2.3}$$

This approximation is good if the parts do not overlap. If the parts overlap each other, this approximation can be bad. For iconic objects (e.g., faces), the prior distribution $P(L|\theta)$ can usually enforce the parts not to overlap, by adopting a distribution that is "rigid" (e.g., in the case of Gaussian distribution, this means small variance of the Gaussian). But for articulated objects (e.g., human body), the parts can easily overlap due to less rigid constraints on the locations of parts. How to handle the overlapping issue properly is a difficult, yet important research problem in human poses estimation. The particular form of $P(I|l_i, u_i)$ depends on the problem domain and features used. Felzenszwalb &

Huttenlocher [30] use silhouette features, but it is also possible to use edge, gradient, or other features.

If the edges E form a tree, the prior distribution $P(L|\theta)$ can be written as:

$$P(L|\theta) \propto \prod_{(v_i,v_j)\in E} P(l_i, l_j|c_{ij}) \tag{2.4}$$

where $P(l_i, l_j|c_{ij})$ is a potential function encoding the prior of the relative location of l_i and l_j. For example, one could use a Gaussian function over the displacement between transformed locations:

$$P(l_i, l_j|c_{ij}) \propto \mathcal{N}\left(T_{ij}(l_i), T_{ji}(l_j), 0, D_{ij}\right) \tag{2.5}$$

where T_{ij}, T_{ij} and D_{ij} are parameters encoded by c_{ij}.

Given the statistical formulation Eq. 2.2, there are several problems need to be solved.

- MAP estimation: finding the optimal L^* that maximize Eq. 2.2, given the image I and the model parameter θ.
- Sampling from posterior: the solution space for pose estimation is usually multi-modal. In general, the best pose L^* does not necessarily solve the whole problem. It is useful to be able to sample from the posterior probability $P(L|I, \theta)$, especially when there are multiple persons in a single image.
- Parameter estimation: this is the problem of learning the model parameters θ from a set of training images. The parameters are usually learned using a maximum likelihood estimation [30]. But recently, other alternative estimations have been proposed, see Sec. 2.1.1.

MAP estimation: the MAP estimation (or equivalently, energy minimization) is to find the optimal configuration defined in Eq. 2.1. If E forms a tree, this problem can be solved by a variant of dynamic programming. The running time of this algorithm is $O(nh^2)$, where n is the number of vertices in the graph (i.e., number of parts), h is the size of all possible locations for each part. In general, h roughly corresponds to the number of pixels in an image, which is prohibitively huge. Felzenszwalb & Huttenlocher [30] show that it is possible to reduce the complexity using a trick called "generalized distance transform", if

$d_{ij}(l_i, l_j)$ is a Mahalanobis distance between transformed locations:

$$d_{ij}(l_i, l_j) = (T_{ij}(l_i) - T_{ji}(l_i))^T \, M_{ij}^{-1} \, (T_{ij}(l_i) - T_{ji}(l_i)) \tag{2.6}$$

where matrix M_{ij} is diagonal, and the transformed locations $T_{ij}(l_i)$ and $T_{ji}(l_j)$ can be represented as positions in a grid.

Sampling from posterior: in the case of tree-structured graphical model, sampling from the posterior distribution $P(L|I, \theta)$ can be done as follows. First, we choose a vertex v_r in the tree as the root and make a directed tree from the graph structure. Then we can calculate the marginal distribution $P(l_r|I, \theta)$ using the Sum-Product algorithm [7]. Given the marginal $P(l_r|I, \theta)$, we can generate a sample for l_r. Then we can traverse the tree from the root to the leaves. At each vertex v_j (with parent v_i), we can calculate the probability $P(l_j|l_i, I, \theta)$, and generate a sample for l_j accordingly. After all the vertices in the tree are traversed, we will obtain a complete sample $L = \{l_i\}_{i \in V}$.

Parameter estimation: suppose we are given a set of m training images $\{I^1, I^2, ..., I^m\}$ and corresponding labeled configurations $\{L^1, L^2, ..., L^m\}$ (we use superscripts to denote image numbers and subscripts to denote part numbers), we can learn the model parameter θ_{ML} using a maximum likelihood criterion:

$$\begin{aligned}
\theta_{ML} &= \arg\max_\theta P(I^1, ..., I^m, L^1, ..., L^m|\theta) \\
&= \arg\max_\theta \prod_{k=1}^{m} P(I^k, L^k|\theta) \\
&= \arg\max_\theta \prod_{k=1}^{m} P(I^k|L^k, \theta) \prod_{k=1}^{m} P(L^k|\theta)
\end{aligned}$$

Since $P(I^k|L^k, \theta)$ and $P(L^k|\theta)$ factorize according to Eq. 2.3 and Eq. 2.4, respectively, it can be shown that we can independently fit the parameters of each factor, i.e., $P(I|l_i, u_i)$ for all $i \in V$, $P(l_i, l_j|c_{ij})$ for all $(v_i, v_j) \in E$.

CRF model

In the traditional pictorial structures, the model parameters θ are estimated by maximizing the joint likelihood of a set of training data:

$$\theta_{ML} = \max_\theta \prod_t P(I^t, L^t|\theta) \tag{2.7}$$

where I^t denotes the t-th training image, and L^t denotes the part locations that have been labeled. However, it has been noticed that the mean pose in a collection of people images tends to have arms lying along the body. Such a pose may not be useful for pose estimation, since the arms tend to be confused with the torso. The ML estimate may not be the best criterion for estimating model parameters, since it is not directly tied to inference. During the inference, we do not need the most likely pose, but the pose that produces the best estimate on the testing image. This motivates the condition random field (CRF) [50] formulation of pose estimation in [83].

In essence, the CRF formulation is very similar to the pictorial structures introduced before. But instead of maximizing the joint likelihood in Eq. 2.7, the CRF formulation tries to estimate the model parameters by maximizing the condition likelihood:

$$\theta_{CL} = \max_{\theta} \prod_{t} P(L^t | I^t, \theta) \tag{2.8}$$

Different from θ_{ML}, θ_{CL} cannot be calculated by closed-form solutions. But it can be estimated by gradient ascent methods. Ramanan & Sminchisescu [83] show that θ_{CL} produces better estimates than θ_{ML}.

Non-tree structured model

Both PS and CRF models assume the human body forms a tree-structured model. Compared with general graphs, tree-structured models have distinct computational advantages in terms of learning and inference. However, tree-structured models have limitations due to their conditional independence assumption: the locations of two parts are independent conditioning on their parent. The tree model simply does not capture the full set of relationships between parts of the body. One example is the color symmetry of clothing, which can be a very useful cue for pose estimation. This information is not captured by the kinematic tree model. Another example is the so-called "double-counting image evidence" problem illustrated in Fig. 2.2. Due to the limitation of tree-structured models, the same piece of image patch can be used twice to explain two different body parts, and the "wrong pose" shown in Fig. 2.2(b) could have a higher posterior probability than the "correct pose" shown in Fig. 2.2(c) under the model.

There has been some work in the literature to address the limitations of tree-structured models. Lan & Huttenlocher [51] propose a common factor model for 2D human pose

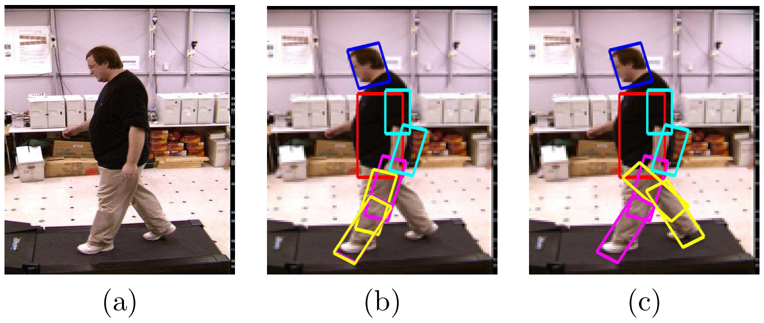


Figure 2.2: Illustration of “double-counting image evidence” problem: (a) original image; (b) wrong pose; (c) correct pose.

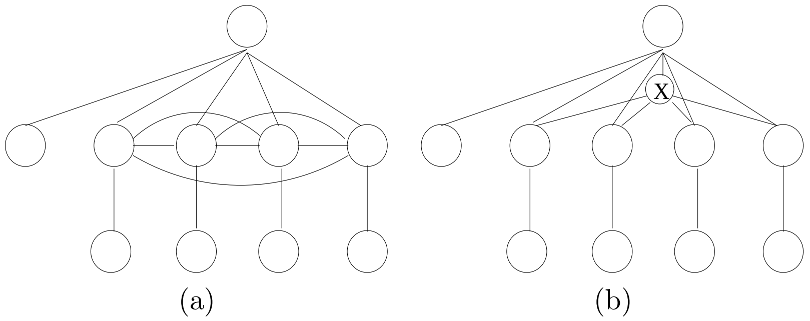


Figure 2.3: Illustration of common-factor model: (a) a kinematic “tree” with additional edges; (b) introducing the common factor variable (labeled “X”) into the model breaks the large clique into several 3-cliques. The figure is from [51].

estimation. It involves two major steps. First, residual covariance analysis is used to identity possible additional spatial relations among parts given the locations of their parents. Those parts whose locations are highly correlated given the locations of their parents are parts that violate the conditional independence assumption of the tree structured model. Additional edges are added to these pairs of nodes to capture their spatial constraints (Fig. 2.3(a)). A problem caused by these additional edges is that they create large cliques in the graph. Since the complexity of learning and inference algorithms in the graphical model is directly related to the size of maximum clique in the graph [7], this means adding these additional edges could potentially makes the algorithm intractable. In order to alleviate this problem, Lan & Huttenlocher augment the original tree with a common factor vertex, which can break the large clique into several 3-cliques (Fig. 2.3(b)).

The approach in Lan & Huttenlocher [51] only works in the situations where we know the activity (e.g., lateral walking) of human figures, it cannot handle human figures in unusual poses. Another way to handle the problem shown in Fig. 2.2 is to explicitly model occlusion relationships between different parts in the model. Sudderth et al. [99] use this idea for hand modeling. Sigal & Black [94] use a similar approach for modeling human poses. The basic idea is to introduce an occlusion variable at each pixel in the image to denotes the probability of that pixel being occluded. Then the image likelihood defined in Eq. 2.3 is modified accordingly, taking into account of these occlusion variables. Although this approach solves the double-counting of image evidence to some extent, there are some limitations as well. First of all, the resulting model becomes extremely complicated, and various approximations have to be made in both learning and inference. Secondly, extra care has to be taken to make sure those occlusion variables are consistent. Neither of [99, 94] provides an easy and principled solution for this problem. Thirdly, due to the complication of the model, both of [99, 94] assume the relative depth ordering of parts is known. It is not clear how this method performs in more general cases.

There are other approaches that explore classes of graph structures that leverage the tradeoff between representational power and computational cost. Song et al. [97] propose the decomposable triangulated graph model (Fig. 2.4) for human motion. Crandall et al. [20] propose k-fan model (Fig. 2.5) for object recognition.

All of the above-mentioned non-tree structured models try to capture additional *spatial* constraints between non-connected body parts. It is also possible to surpass the limitations of tree models with respect to the color symmetry cue. One such example is Ramanan [78]. Although technically the graphical model used in [78] is still tree-structured, it achieves representational power beyond trees, so we will list it here. The method in Ramanan [78] proceeds by firstly running a tree-structure CRF model on a test image. Then it samples candidate locations from the posterior, builds a color histogram for each model part using the color information from these candidate part locations, and runs the color histogram model to re-infer the part locations. The color symmetry can be enforced by learning a single color histogram model for the left/right limbs.

2.1.2 Bottom-up approach

The search space of pose configurations is usually extremely huge. The global optimization approach tries to find the optimal configuration by essentially evaluating all possible solutions in the whole space. Those approaches usually have to make some particular assumptions about the problem formulation (e.g., the particular form of $d_{ij}(l_i, l_j)$ in PS)

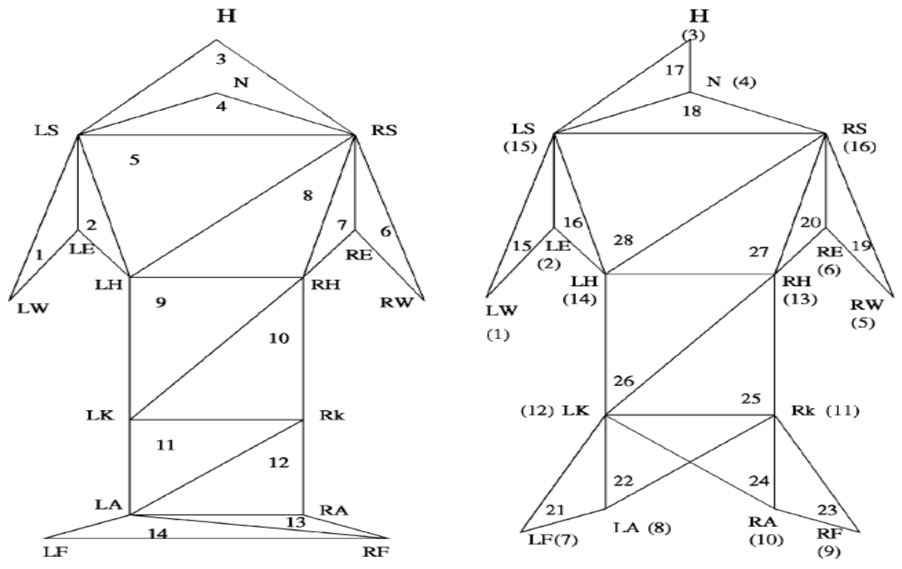


Figure 2.4: Two examples of decomposable triangulated graphs for human body. This figure is from [97].

in order to make the algorithm computationally feasible by using various computational tricks (e.g., distance transform [30], convolution [20, 83, 78]).

Alternatively, we can adopt a bottom-up approach by pruning unreasonably configurations early on. These approaches usually first detect candidate parts in the image, then combine them into a valid full-body configuration.

Candidates from segmentation: one possible way to find candidate body parts is through image segmentation. It is observed in Mori et al. [68] that salient half-limbs often pop out as a single segment in image segmentation, as shown in Fig. 2.6. Mori et al. [68] learn

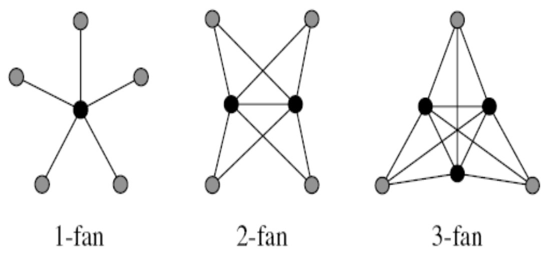


Figure 2.5: Examples of k-fan models. This figure is from [20].

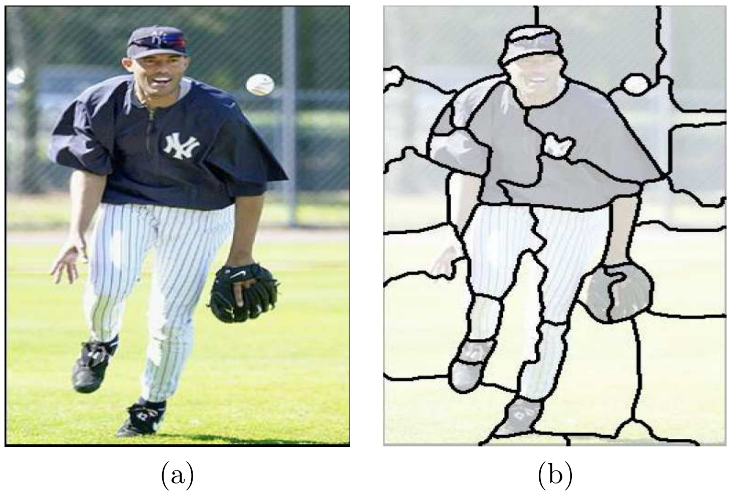


Figure 2.6: Illustration of using image segmentation for finding candidate limbs: (a) original image; (b) result of Normalized Cuts segmentation [93] with 40 segments. Note some of the segments correspond to half-limbs. This figure is from [68].

a classifier to find half-limbs and the torso from these segments, and try to assemble them together. Mori [65] uses a representation called "superpixels" (Fig. 2.7), which essentially are over-segmentations of the image. The joints of half-limbs are approximated to be the centers of these superpixels, and the half-limbs are assumed to be composed of superpixels. Then these superpixels are assembled together to form half-limbs, and further to whole body configurations.

Candidates from parallel lines: body parts in the 2D image can also be approximated by parallel line segments. Ioffe & Forsyth [43] and Ren et al. [84] find parallel line segments as candidate parts in an image, then try to assemble them into full body configuration. Similarly, Ramanan & Forsyth [80] find parallel line segments in a video sequence as candidate parts, then try to assemble a full body from them by exploiting both temporal and spatial coherence.

Bottom-up approaches can be applied in an incremental, hierarchical fashion. Ioffe & Forsyth [43] use a hierarchical pyramid of classifiers shown in Fig. 2.8, where each node in the pyramid corresponds to a classifier. The classifiers at the leaves take all the parallel line segments as inputs. All the other nodes take the outputs of their children as the inputs. The root node output the complete assemblies of human body.

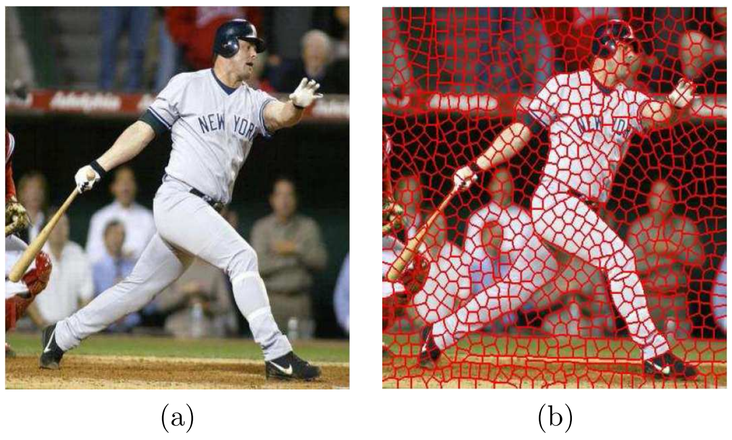


Figure 2.7: Examples of "superpixels": (a) original image; (b) superpixel representation created by over-segmentation. This figure is from [65].

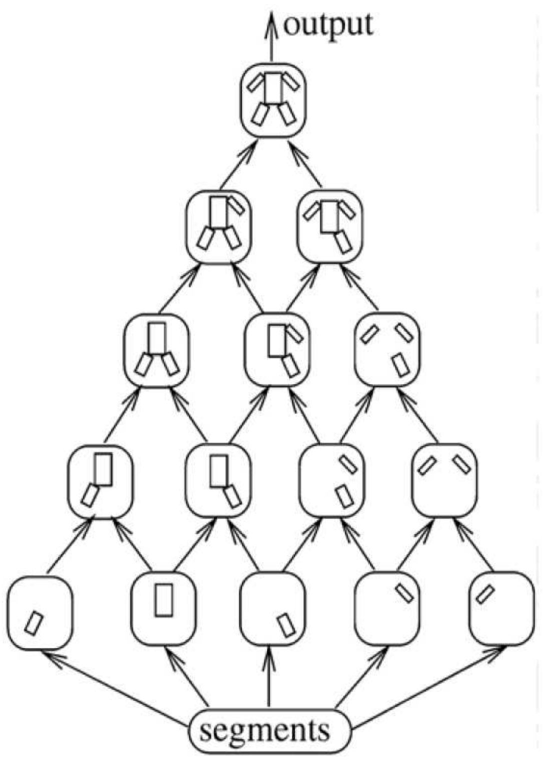


Figure 2.8: A pyramid of classifiers used in [43]. This figure is from [43].

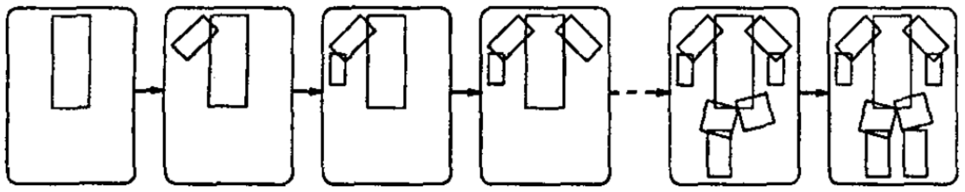


Figure 2.9: Sampling assemblies $S_1, S_2, ..., S_9$ incrementally. This figure is from [44].

2.1.3 Sampling-based approaches

The third class of approaches explore the solution space of human poses using sampling-based methods. The basic idea is to approximate the whole space of human poses by a finite number of "particles". Examples of such approaches include Ioffe & Forsyth [43], Sigal & Black [94], etc.

In Ioffe & Forsyth [43], the image likelihood for a nine segment assembly is represented by a set of 41 geometric features $f_i (i = 1, 2, ..., 41)$. These features are chosen in a way so that they can be assumed to be independent. Then the image likelihood of an assembly A containing nine body segments, the likelihood of A can be written as $L(A) = \prod_{i=1}^{41} d_i(f_i)$, where $d_i(f_i)$ is the corresponding one-dimensional marginal likelihood. Fixing a permutation $(l_1, ..., l_9)$ of labels (e.g., $\{T, LUA, ...\}$), Ioffe & Forsyth [43] incrementally generate a sequence $(S_1, S_2, ..., S_9)$, where S_k represents sample assemblies with k segments $(s_{l_1}, s_{l_2}, ..., s_{l_k})$. For example, S_1 contains samples s_T of {torso} segment, while S_2 contains samples of (s_T, s_{LUA}) of {torso, left upper arm} segments. S_9 will contain samples of all the nine segments (Fig. 2.9). The samples in S_k are drawn from the marginal likelihood $L_{l_1 ... l_k}(A) = \prod_i d_i(f_i)$, where the product is over all the features computable from segments labeled $l_1, ..., l_k$.

Importance sampling is used to generate S_{k+1} from samples in S_k. Firstly, the set of sub-assemblies $(s_{l_1}, s_{l_2}, ..., s_{l_k})$ for all groups $(s_{l_1}, s_{l_2}, ..., s_{l_k}) \in S_k$ and all choices of $s_{l_{k+1}}$ are formed. The first component is a sample from the relevant marginal distribution, but the second component is not. A re-sampling step is used to correct this bias. It proceeds by independently drawing N samples, with probability of drawing $(s_{l_1}, s_{l_2}, ..., s_{l_k})$ proportional to $w(s_{l_1}, s_{l_2}, ..., s_{l_k}) = \frac{L_{l_1 ... l_{k+1}}(\cdot)}{L_{l_1 ... l_k}} = \prod_i d_i(f_i)$, where the product is over all features that depend on $s_{l_{k+1}}$ and possibly some of $s_{l_1}, ..., s_{l_k}$.

Sigal & Black [94] use sampling within the belief propagation inference. In their work, the configuration of the body parts is represented by a set of latent variables $\mathcal{X} = \{X_1, X_2, ..., X_M\}$.

The joint probability of $\{X_1, X_2, ..., X_M\}$ is defined by a pairwise Markov Random Field (MRF) as $P(\mathcal{X}) = \frac{1}{Z}\left(\prod_{(i,j)\in E}\psi_{ij}(x_i, x_j)\right)(\prod_i \phi_i(x_i))$, where E denotes the set of edges in MRF. Solving the MRF using belief propagation involves passing messages from x_i to x_j iteratively as follows:

$$m_{ij}(x_j) \leftarrow \int_{x_i} \psi_{ij}(x_j, x_i)\phi_i(x_i) \prod_{k\in\mathcal{N}(i)\setminus j} m_{ki}(x_i)dx_i \tag{2.9}$$

where $\mathcal{N}(i)\setminus j$ denotes the set of neighbors of x_i excluding x_j.

The integration in Eq. 2.9 is usually intractable if x_i is a continuous random variable. Sigal & Black [94] use a Monte-Carlo integration by drawing N particles $s_{ij}^n \sim g_{ij}(\cdot)$, where $g_{ij}(x_i)$ is a so-called *importance function*. Then the message $m_{ij}(x_j)$ can be approximated by these particles as:

$$\tilde{m}_{ij}(x_j) = \frac{1}{\sum_{n=1}^N \pi_{ij}^n} \sum_{n=1}^N \pi_{ij}^n \psi_{ij}(x_j, s_{ij}^n) \tag{2.10}$$

where $\pi_{ij}^n \propto p_{ij}^M(s_{ij}^n)/g_{ij}(s_{ij}^n)$ is the importance weight of the n-th particle s_n, and $p_{ij}^M = \frac{1}{Z_{ij}}\phi_i(x_i)\prod_{k\in\mathcal{N}(i)\setminus j} m_{ki}(x_i)$.

Eq. 2.10 essentially provides a way of representing the continuous message $m_{ij}(x_j)$ by a discrete set of particles. And the belief propagation in the discrete case can be easily computed.

There are approaches that use the output of low-level detectors to construct a good importance function for drawing samples. For certain body parts (e.g., faces, skin), there are relatively good low-level detectors available. These approaches try to incorporate the information provided from these low-level detectors. For example, Lee & Cohen [58] develop a MCMC approach for inferring 3D human poses, where the proposal distribution in the MCMC algorithm is guided by the low-level detectors. Hua et al. [39] propose a similar approach for inferring 2D human poses.

2.1.4 Model-free Approaches

Model-free approaches for human pose estimation do not make any assumptions about any particular forms on the distributions of possible poses. Instead, they try to learn the mapping directly from the image features to poses. In this chapter, we review different kinds of features commonly used in model-free approaches, and two major classes of model-free learning methods.

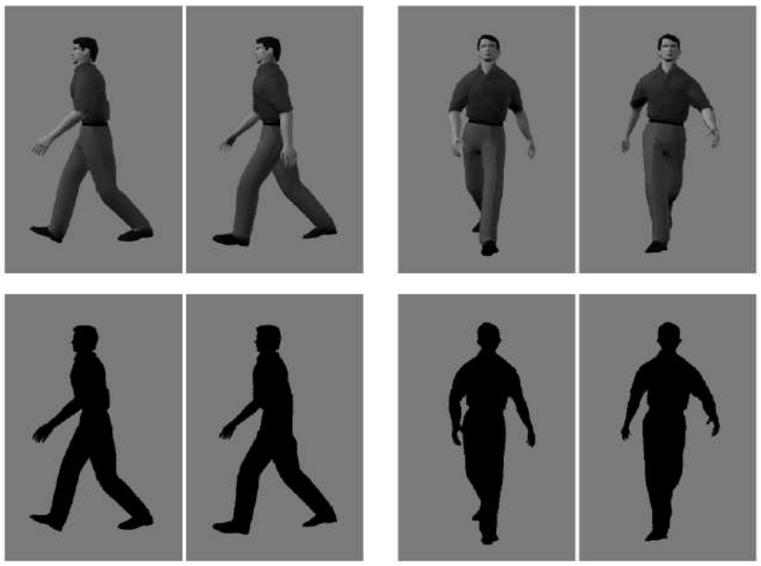


Figure 2.10: Examples of silhouette features. Note in the first two examples, the silhouette features are similar, and cannot tell whether the person is stepping on left or right leg. This figure is from [2].

Features

Some of the work [2, 17, 86] in the literature use features based on image silhouettes. Silhouette features encode some rich information about the pose, are insensitive to clothing color, textures, etc. These features are be extracted from images when robust background subtraction or motion-based segmentation is available. But they also have some limitations. First of all, silhouette features do not have any information about the "interior" of the human. So they cannot distinguish frontal views from back views, whether a person seen from the side is stepping the left leg or the right one (Fig 2.10) [2].

Based on the binary silhouette of a human figure, different feature vectors can be extracted. For example, Agarwal & Triggs [2] use shape contexts [4] on top of the silhouette. Given a set of points sampled along the silhouette, the shape context for a point is a histogram of local silhouette boundary pixels into log-polar bins. A silhouette is characterized by the distribution of these shape contexts. Agarwal & Triggs [2] also uses a second-level histogramming that vector quantize the shape context space and use this to reduce the distribution of each silhouette to a 100D histogram (Fig. 2.12). This is similar to the idea of "shapeme" [66] used in shape context matching.

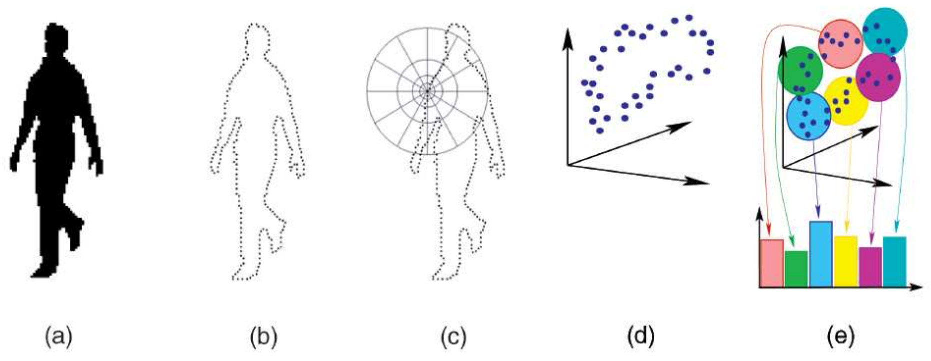


Figure 2.11: Illustration of the silhouette based feature used in [2]: (a) input silhouette; (b) sampled edge points; (c) local shape contexts computed on edge points; (d) distribution in shape context space; (e) vector quantization of the distribution into a 100D histogram. This figure is from [2].

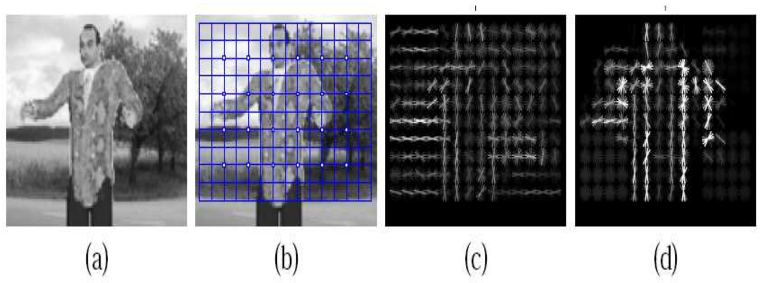


Figure 2.12: Illustration of the gradient-feature in [1]. (a) original image; (b) a grid of cells where descriptors are computed; (c) descriptors computed at these cells; (d) suppressing background using NMF bases. This figure is from [1].

Due to the limitations of silhouette features, some people have proposed to use edge or gradient based features for pose estimation. Mori & Malik [67] use shape context descriptors based on sampled edge points. Shakhnarovich [91] use histograms of multi-scale edge directions. Edges are detected using Sobel operator and each edge pixel is classified into one of four direction bins. Then the histograms of direction bins are computed within sliding windows of varying sizes placed at multiple locations in the image. The concatenation of all the histograms becomes the feature vector for the image. Agarwal & Triggs [1] use a similar feature based on image gradients, where they compute histograms of gradient orientations in small spatial cells. They also use an additional step of non-negative matrix factorization (NMF) bases to suppress the gradients in the background (Fig. 2.12).

2.1.5 Learning methods

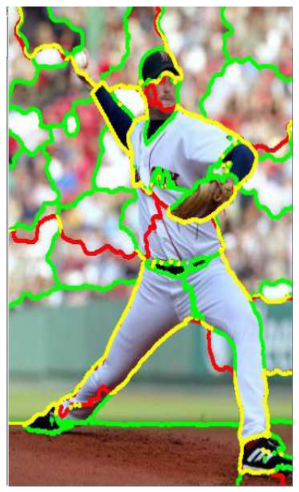
Given a feature vector x extracted from an image, model-free pose estimate approaches try to learn a mapping from x to the pose parameter vector θ. There are two main classes of learning methods used in the literature: exemplar-based methods (Sec.2.1.5) and regression-based methods (Sec.2.1.5).

Exemplar-based methods

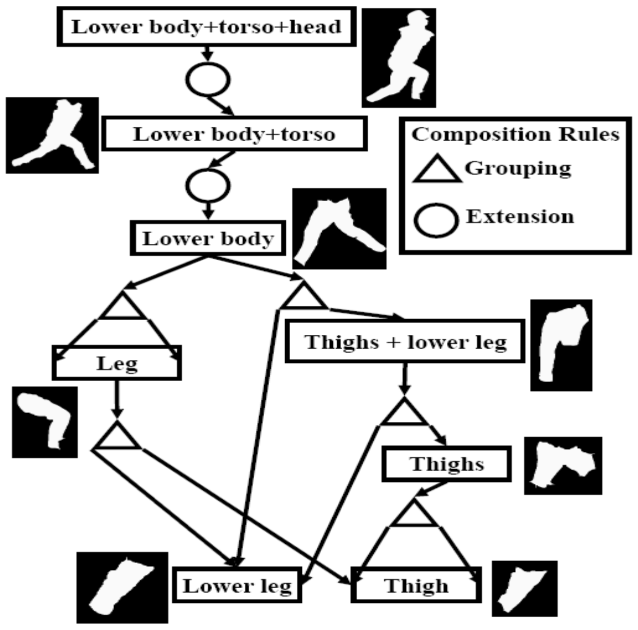
The first class of approaches are based on exemplar matching. These approaches require a database of training images $\{x_1, x_2, ..., x_n\}$ with corresponding labeled poses (e.g., joints, half-limbs, etc) denotes as $\{\theta_1, \theta_2, ..., \theta_n\}$. For a new test image x', exemplar-based methods firstly find one or more images from the training dataset that are most similar to x', based on certain distance measurement. Then the pose for the new image x' is obtained by transferring the body poses from those matched images. Mori & Malik [67] and Sullivan & Carlsson [100] use exemplar matching for 2D pose estimation. They try to match the test image to the best candidate in the training images, then warp the candidate image to more closely match the test image.

Exemplar matching approaches obviously require a large database of training images. When the number of training images is too large, searching the best candidate can be slow. To address this issue, Shakhnarovich [91] adopt parameter-sensitive hashing (PSH) in exemplar matching. The basic idea is to build a hash table for the training images, such that the probability of collision is large for examples similar in their parameters and small for dissimilar ones. Then we can perform approximate matching using this hash table for speedup.

Since human figures are highly articulated, matching a whole image of a human figure is challenging due to the range of deformations. Srinivasan & Shi [98] propose a exemplar matching approach combined with image segmentation. This approach is based on the following assumption: matching a whole human figure is usually difficult (e.g., the test image might not be similar enough to any of the training images), but matching partial bodies is relatively easier. In their approach, candidate partial bodies are obtains by multiple segmentations of an input image (Fig. 2.13(a)). Their method recursively groups bottom-up body parts into increasingly larger parts until the whole body is formed. The grouping process is guided by a parse tree (Fig. 2.13(b)). Each node in the parse tree corresponds to



(a)



(b)

Figure 2.13: Illustration of the idea of multi-segmentation and parse tree in [98]: (a) two segmentations of an image; (b) body parse tree shown with an exemplar shape from the training set for each node. This figure is from [98].

a partial body part. The structure of the tree and a set of parse rules define how partial body parts can be grouped together. At each node of the parse tree, an exemplar matching method is used to score how well the partial body matches to an exemplar partial body in the training set. The best pose estimation is obtained by combining the matching scores of the whole parse tree.

Exemplar-based methods can also be incorporated into a probabilistic framework. Toyama & Blake [105] propose a probabilistic tracking system using exemplars. They select exemplars as representatives of raw training data, then use them to represent probabilistic mixture distributions of object configurations. The advantage of using exemplars in this framework is that hand-construction of object models is avoided.

Regression-based methods

Whereas exemplar matching approaches infer the pose for an image by directly finding the most similar training image, regression-based methods first learn a smoothed regression function $f(x) \to \theta$ that directly maps the image feature vector x to the pose parameter vector θ, then use the learned regression function $f(\cdot)$ to infer the pose θ given a new image x.

Agarwal & Triggs [2] use ridge regression and relevance vector machines (RVM) for 3D human pose estimation. Ridge regression is a standard regression technique that introduces a regularization term in the least square regression. RVM is a sparse Bayesian technique similar to support vector machines (SVM), but RVM is for regression, and SVM is for classification.

2.2 Human Action Recognition

Recognizing human actions from image sequences is a challenging problem in computer vision. It has applications in many areas, e.g., motion capture, medical bio-mechanical analysis, ergonomic analysis, human-computer interaction, surveillance and security, environmental control and monitoring, sport and entertainment analysis, etc. A lot of work has been done in recognizing actions from both still images and video sequences. In this review, we focus on recognition based on motion cues, although we are aware that other cues (e.g., shape cues [100, 61, 104]) have also been used for action recognition.

2.2.1 Motion-based Action Recognition

There are many approaches that perform action recognition by analyzing patterns of motion. For example, Cutler & Davis [21], and Polana & Nelson [74] detect and classify periodic motions. Little & Boyd [59] analyze the periodic structure of optical flow patterns for gait recognition. Efros et al. [24] classify actions by matching motion features based on optical flows. There is also work using both motion and shape cues. For example, Bobick & Davis [13] use a representation known as “temporal templates” to capture both motion and shape, represented as evolving silhouettes. Shechtman & Irani [92] propose a space-time correlation method that can detect similarity between video segments. Jhuang et al. [45] apply neurobiological model of motion processing for action recognition using space-time gradient and optical flow features. Schindler & Van Gool[88] perform action recognition by training SVM classifiers based on local shapes and dense optical flows. Rodriguez et al.[85] propose a template-based method based on a maximum average correlation height filter that is capable of capturing intra-class variabilities.

2.2.2 Temporal Dynamic Models

There is a line of work on building sophisticated temporal dynamic models for modeling and understanding activities. The early work of Yamato et al. [119] uses the hidden Markov model (HMM) for recognizing human actions from image sequences. Feng and Perona [31] build HMM models from vector-quantized image shapes called “movelets”. Olivera et al. [72] use a layered HMM to represent office activities. Recently, Xiang & Gong [118] model complex activities of multiple objects in cluttered scenes using dynamic Bayesian networks. Ikizler and Forsyth [42] use finite state models to search for complex activities from videos. Laxton et al. [55] build a dynamic Bayesian network to leverage temporal, contextual and ordering constraints for recognizing complex activities in video.

Although generative models (e.g., HMM) are popular in sequence modeling, recently people have also been applying discriminative models for modeling temporal information, e.g., conditional random fields (CRF) in Sminchisescu et al. [96] and hidden conditional random fields (HCRF) in Wang et al. [110].

A drawback of these models is that they have to make some assumptions (e.g., the independence assumption made by 1-st order HMM) in order to be computationally tractable. It is also hard to learn these models since there are usually many model parameters to be set.

2.2.3 Action Recognition with Tracking

Another line of work is to first track the body parts and then perform action recognition based on the motion trajectories. Ramanan & Forsyth[79] introduce a system for automatic annotation of everyday movements. They first track the limbs of a human in 2D using a pictorial structure model. Then they synthesize 3D motion sequence which looks like the 2D track by lifting the track to 3D and matching them to a library of annotated motion capture data. The annotation for the underlying video sequence is obtained from the annotations associated with the synthesized 3D motion sequence. Yilmaz & Shah [120] first obtain a set of landmarks on the human body by manual labeling, then perform action recognition by analyzing the corresponding 4D (x, y, z, t) trajectories of the landmarks. Song et al. [97] and Fanti et al. [25] first track and detect feature point in each frame of a video, then obtain various cues (e.g., position, velocity) for learning the human motion.

2.2.4 Interest Point Methods

A popular approach in action recognition is based on spatial-temporal interest points and local feature descriptors. For convenience, those local descriptors are usually vector-quantized to obtain a finite set of "visual words" before they are fed into any classification algorithms. Laptev and Lindeberg [52] propose a space-time interest point operator that detect local structures in space-time that image observations have large local variations in both space and time. Schuldt et al.[90] train an SVM classifier based on this space-time features for recognizing human actions. Dollár et al. [23] propose space-time interest point detector based on a set of linear filters, and use these local features with k-nearest neighbor classifier for action recognition. Ke et al. [47] build a cascade of classifiers based on space-time volumetric features for event detection. Nowozin et al. [71] first detect local interest points, then learn a set of discriminative subsequences for action classification by exploiting the sequence mining techniques from data mining. Liu & Shah[60] exploit mutual information maximization techniques to learn a compact set of visual words. Niebles & Fei-Fei[69] combine shape information with local appearance features by building a hierarchical model that can be characterized as a constellation of bag-of-features. Local descriptors extracted from space-time interest points have also been shown to work well on videos with complex scenes (e.g., movies), e.g., Laptev & Pérez [54] and Laptev et al. [53] learn a boosted classifier based on those local descriptors to do action recognition on movie data.

Chapter 3

Human Pose Estimation

As mentioned earlier, estimating human body poses from still images is a very challenging computer vision problem. In order to reliably interpret still images of human figures, it is likely that multiple cues relating different parts of the figure will need to be exploited.

Many existing approaches to this problem model the human body as a combination of rigid parts, connected together in some fashion. The typical configuration constraints used are kinematic constraints between adjacent parts, such as torso-upper half-limb connection, or upper-lower half-limb connection. This set of constraints has a distinct computational advantage – since the constraints form a tree-structured model, inferring the optimal pose of the person using this model is tractable.

However, this computational advantage comes at a cost. Simply put, the single tree model does not adequately model the full set of relationships between parts of the body. Relationships between parts not connected in the kinematic tree cannot be directly captured by this model.

The main contribution of the work presented in this chapter is a framework for modeling human figures as a collection of trees. We argue that this framework has the advantage of being able to locally capture constraints between the parts which constitute the model. With a collection of trees, a global set of constraints can be modeled. We demonstrate that the computational advantages of tree-structured models can be kept, and provide tractable algorithms for learning and inference in these multiple tree models. We present two applications of our framework. The first application uses the multiple tree model framework for occlusion reasoning. The second application combines multiple deformable trees to capture a richer set of spatial constraints between body parts. We also provide an analysis of our approach which compares to existing approaches for combining multiple trees [63, 109].

The rest of this chapter is organized as follows. Section 3.1 introduces how to model the human body using a single tree-based model. Section 3.2 describes our multiple tree approach and how to apply it to model spatial constraints beyond those captured by a single-tree model. Section 3.3 describes how to apply this multiple tree approach for modeling occlusion relationships between body parts. Section 3.4 presents our experimental results. Section 3.5 concludes this chapter.

3.1 Modeling the Human Body

In our method we use a combination of tree-structured deformable models for human pose estimation. The basic idea is to model a human figure as a weighted combination of several tree-structured deformable models. The parameters of each tree model are learned from training data in a discriminative fashion.

We first describe how we model the human body, and then demonstrate how this model of multiple trees can be used for modeling spatial constraints and occlusion reasoning. We also relate the pictorial structures [30] defined with pixel likelihoods and the CRF model [78] defined with patch likelihoods. This connection turns out to be useful when we develop our occlusion reasoning scheme in Section 3.3.

3.1.1 Single Tree-structured Deformable Body Models

Consider a human body model with K parts, where each part is represented by an oriented rectangle with fixed size. We can construct an undirected graph $G = (V, E)$ to represent the K parts (Fig. 3.1(a)). Each part is represented by a vertex $v_i \in V$, and there exists an undirected edge $e_{ij} = (v_i, v_j) \in E$ between vertices v_i and v_j if v_i and v_j has some dependency. Let $l_i = (x_i, y_i, \theta_i)$ be a random variable encoding the image position and orientation of the i-th part, we denote the configuration of the K part model as $L = (l_1, l_2, ..., l_K)$. Given the model parameters Θ and assuming no occlusions, the conditional probability of L in an image I can be written as:

$$P(L|I,\Theta) \quad \propto \quad P(L|\Theta)P(I|L,\Theta) = P(L|\alpha)\prod_i P(I|l_i,\beta_i) \tag{3.1}$$

where we explicitly decompose $P(L|I,\Theta)$ into the prior term $P(L|\alpha)$ and the product of several likelihood terms $P(I|l_i,\beta_i)$. Each $P(I|l_i,\beta_i)$ is a local likelihood for the part i.

Assuming pixel independence, we can write each local likelihood as:

$$P(I|l_i,\beta_i) = \prod_{u\in\Omega(l_i)} P_{l_i(u)}\left(f(I_u)\right) \prod_{\Upsilon-\Omega(l_i)} P_{bg(u)}\left(f(I_u)\right) \propto \prod_{u\in\Omega(l_i)} \frac{P_{l_i(u)}(f(I_u))}{P_{bg(u)}(f(I_u))} \tag{3.2}$$

In the above equation, we have used the following notation. u is a pixel location in the image I, I_u is the image evidence at pixel u. In our work, we use binary edges as our image evidences. $f(I_u)$ is a function returning 1 if I_u is an edge point, and 0 otherwise. $\Omega(l_i)$ is the set of pixels enclosed by part i as defined by l_i. Υ is the set of all pixels in the whole image. $P_{l_i(u)}$ is a binomial distribution indicating how likely the pixel u is an edge point under the part l_i. $P_{bg(u)}$ is a binomial distribution for the background.

Let $\frac{P_{l_i(u)}(f(I_u))}{P_{bg(u)}(f(I_u))} = \exp(\beta_{i(u)}f(I_u))$, we will have:

$$\begin{aligned} P(I|l_i,\beta_i) &\propto \prod_{u\in\Omega(l_i)} \exp\left(\beta_{i(u)}f(I_u)\right) = \exp\left(\sum_{u\in\Omega(l_i)} \beta_{i(u)}f(I_u)\right) && (3.3)\\ &= \exp\left(\beta_i^T f_i(I(l_i))\right) && (3.4) \end{aligned}$$

where $f_i(I(l_i))$ is the part-specific feature vector extracted from the oriented image patch at location l_i. In our case, it is a binary vector of edges for part i. β_i is a part-specific parameter that favors certain edge patterns for an oriented rectangle patch $I(l_i)$ in image I. In our formulation, β_i is simply the concatenation of $\{\beta_{i(u)}\}_{u\in\Omega(l_i)}$. We visualize β_i in Fig. 3.1(d).

Following previous work [78], we assume the prior term $P(L|\alpha)$ is defined in terms of the relative locations of the parts as follows:

$$P(L|\alpha) \propto \exp\left(\sum_{(i,j)\in E} \psi(l_i - l_j)\right) \tag{3.5}$$

Most previous approaches use Gaussian shape priors $\psi(l_i - l_j) \propto \mathcal{N}(l_i - l_j; \mu_i, \Sigma_i)$ [30]. However, since we are dealing with images with a wide range of poses and aspects, Gaussian shape priors seem too rigid. Instead we choose a spatial prior using discrete binning (Fig. 3.1(c)) similar to the one used in Ramanan [78]:

$$\psi(l_i - l_j) = \alpha_i^T \text{bin}(l_i - l_j) \tag{3.6}$$

bin$(\cdot)$ is a vector of all zeros with a single one for the occupied bin. α_i is a parameter that favors certain spatial and angular bins for part i with respect to its parent j. This spatial prior captures more intricate distributions than a Gaussian prior.

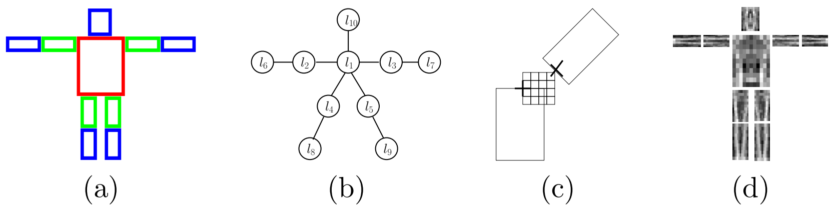


Figure 3.1: Representation of a human body: (a) human body represented as a 10-part model; (b) corresponding kinematic tree structured model; (c) discrete binning for spatial prior; (d) visualization of the learned edge-based appearance model β_i for each body part. Dark areas correspond to small values of β_i, and bright areas correspond to large values of β_i.

Combining (3.1), (3.4) and (3.5), we obtain the following formulation:

$$P(L|I,\Theta) \propto \exp\left(\sum_{(i,j)\in E} \psi(l_i - l_j) + \sum_{i=1}^{K} \phi(l_i)\right) \tag{3.7}$$

where $\phi(l_i)$ is a potential function that models the local image evidence for part i located at l_i. $\phi(l_i)$ is defined as $\phi(l_i) = \beta_i^T f_i(I(l_i))$.

Equation (3.7) is exactly the same Conditional Random Field (CRF) formulation of human pose estimation problem in Ramanan [78]. This shows the two different formulations (pictorial structures [30] and CRF [78]) of human pose estimation problems are in fact equivalent. The only difference is that they use different criteria for model parameter learning, i.e., maximizing the joint likelihood (ML) or the conditional likelihood (CL). In the following, we will use the CRF formulation in (3.7). But we will come back to the pictorial structure formulation of (3.1), when we develop our occlusion reasoning method.

To facilitate tractable learning and inference, G is usually assumed to form a tree $T = (V, E_T)$ [30, 78]. In particular, most work uses the kinematic tree (Fig. 3.1(b)) as the underlying tree model.

Inference in a single tree-structured model can be done by message-passing. Using 3D convolution, one can search exhaustively over all part locations in an image without relying on feature detectors. Learning of the model parameters can be done by closed-form solutions (ML) or gradient ascent methods (CL). See [78, 111] for details.

3.1.2 Multiple Tree Models

In our work we model the human body by a collection of multiple tree-structured models. For example, one can use the two tree models in Fig. 3.4 to model the kinematic constraints and the occlusion relationships between the legs. One can also use different tree structures (e.g., Fig. 3.6) to model the spatial constraints that are not captured in the kinematic tree. The weighting parameters which combine the multiple tree models can also be learned in a discriminative fashion using boosting (Section 3.2).

The final form of our model is:

$$F(L, I; \Theta) = \sum_t w_t f_t(L, I; \Theta) \tag{3.8}$$

where $f_t(L, I; \Theta)$ is a single tree model with tree structure τ_t, w_t is the weight associated with this single tree model. The optimal pose L^* can be obtained in our model as $L^* = \arg\max_L F(L, I; \Theta)$. In the next section, we describe the algorithm for learning all the model parameters $\Theta = \{w_t, \Theta_t\}$.

3.2 Spatial Constraints with Multiple Trees

Our learning algorithm for spatial constraints is based on AdaBoost.MRF proposed in Truyen et al. [107]. Given an image I, the problem of pose estimation is to find the best part labeling L^* that maximizes $F(L, I)$, i.e. $L^* = \arg\max_L F(L, I)$. $F(L, I)$ is known as the "strong learner" in the boosting literature. Given a set of training examples $(I^i, L^i), i = 1, 2, ..., N$. $F(L, I)$ is found by minimizing the following loss function L_O:

$$L_O = \sum_i \sum_L \exp\left(F(I^i, L) - F(I^i, L^i)\right) \tag{3.9}$$

We assume $F(L, I)$ is a linear combination of a set of so-called "weak learners", i.e., $F(I, L) = \sum_t w_t f_t(L, I)$. The t-th weak learner $f_t(L, I)$ and its corresponding weight w_t are found by minimizing the loss function defined above, i.e. $(f_t, w_t) = \arg\max_{f,w} L_O$. In our case, we choose the weak learner as $f(L, I) = \log p(L|I)$. To achieve computational tractability, we assume each weak learner is defined on a tree model.

If we can successfully learn a set of tree-based weak learners $f_t(L, I)$ and their weights w_t, the combination of these weak learners captures more dependencies than a single tree

```
**Input:** $i = 1, 2, ..., D$ data pairs, graphs $\{G_i = (V_i, E_i)\}$
**Output:** set of trees with learned parameters and weights
Select a set of spanning trees $\{\tau\}$
Choose the number of boosting iterations $T$
Initialize $\{w_{i,0} = \frac{1}{D}\}$, and $w_1 = 1$
**for** each boosting round $t = 1, 2, ..., T$
    Select a spanning tree $\tau_t$
    /* Add a weak learner */
    $\Theta_t = \arg\max_\Theta \sum_i w_{i,t-1} \log P_{\tau_t}(L_i, I_i|\Theta)$
    $f_t = \log P_{\tau_t}(L|I, \Theta_t)$
    **if** $t > 1$ **then**
        select the step size $0 < w_t < 1$ using line searches
    **end if**
    /* Update the strong learner */
    $F_t = \frac{1}{1+w_t} F_{t-1} + \frac{w_t}{1+w_t} f_t$
    /* Scale down the previous learners' weights*/
    $w_j \leftarrow \frac{w_j}{1+w_t}$, for $j = 1, 2, ..., t$
    /* Re-weight training data*/
    $w_{i,t} \propto w_{i,t-1} \exp(-w_t f_{i,t})$
**end for**
Output $\{\tau_t\}$,$\{\Theta_t\}$ and $\{w_t\}$, $t = 1, 2, ..., T$
```

Figure 3.2: Algorithm of boosted multiple trees

model. At the same time, the inference in this model is still tractable, since each component is a tree.

Optimizing L_O is difficult, Truyen et al. [107] suggest optimizing the following alternative loss function: $L_H = \sum_i \exp\left(-F(L^i, I^i)\right)$. It can be shown that L_H is an upper bound of the original loss function L_O, provided that we can make sure $\sum_j w_j = 1$. In Truyen et al. [107], the requirement $\sum_j w_j = 1$ is met by scaling down each previous weak learner's weight by a factor of $1 - w_t$ as $w_j^{'} \leftarrow w_j(1 - w_t)$, for $j = 1, 2, ..., t-1$, so that $\sum_{j=1}^{t-1} w_j^{'} + w_t = \sum_{j=1}^{t-1} w_j(1 - w_t) + w_t = 1$, since $\sum_{j=1}^{t-1} w_j = 1$. In practice, we find that this trick sometimes scales down previous weak learners to have zero weights. So we use a slightly different method by scaling down each weak learner's weight up to t by a factor of $1/(1 + w_t)$. It can be shown that we still have $\sum_{j=1}^{t} w_j = 1$. Figure 3.2 shows the overall algorithm.

Discussion: Our model is similar to "mixtures of trees" (MoT) [63] at a first glance, but there are some important differences. MoT is a generative model developed for density

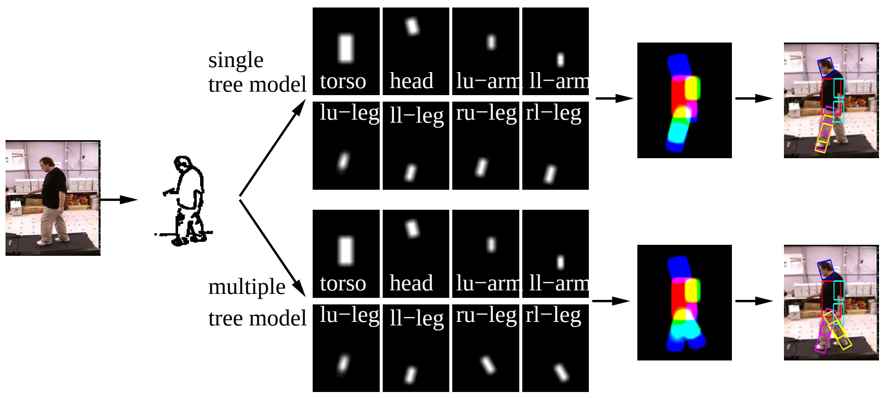


Figure 3.3: Illustration of "double counting of image evidence" problem: top row shows how the same piece of image patch is used to explain two body parts, the bottom row shows how our occlusion reasoning mechanism using multiple trees can alleviate this problem.

modeling problem. It is not designed for classification or prediction. Although one can use MoT as the spatial prior in a generative fashion, it is not clear how to learn the model in a discriminative way. Instead, our model is trained discriminatively, and our objective function is more closely tied to inference.

Another similar work is the tree-reweighted message passing (TRW)[109]. TRW aims to approximate the partition function in MRF, it does not answer the question of learning a good model for recognition, i.e., TRW assumes the MRF model is given, and it simply tries to solve the inference problem. Plus, TRW is an iterative algorithm, and its convergence is still an unsolved problem.

3.3 Occlusion Reasoning with Multiple Trees

In this section, we apply the multiple tree framework to the "double counting of image evidence" problem in human pose estimation illustrated in the top row of Fig. 3.3, where the same image patch is used twice to explain two different body parts. Previous approaches [51] have focused on using strong priors of body poses to solve this problem. However, these approaches are limited to the cases of normal poses and known activities. We believe the proper way to solve this problem is to introduce occlusion reasoning in the model. In our multiple tree framework, we can define one tree for the kinematic constraint (e.g., Fig. 3.4(a)), and a second tree for the occlusion relationships (e.g., Fig. 3.4(b)). In this section, we discuss how to incorporate occlusion reasoning into the human body model introduced in Section 3.1, and how to do inference in a tree model involving occlusion relationships (see Fig. 3.4(b)). Before we proceed, we first clarify the terminology we are using. By "occlusion reasoning", we do not necessarily mean the body parts in the image are occluding each other, instead we use "occlusion" to refer to the particular problem of using the same image patch to explain different body parts, as illustrated in Fig. 3.3.

Occlusion-sensitive formulation: The factorization of the global likelihood into local likelihood terms in Eq. 3.1 is valid only if the local terms $P(I|l_i, \beta_i)$ for $i \in \{1..K\}$ are independent. This assumption holds when there are no occlusions among different parts. In order to obtain a similar decomposition (hence distributed inference) when occlusions exist, we augment the configuration l_i of part i with a set of binary hidden variables $z_i = \{z_{i(u)}\}_{u\in\Upsilon}$, similar to [99]. Note that there is a binary variable $z_{i(u)}$ for each *pixel*. Let $z_{i(u)} = 0$ if pixel u in the area enclosed by part i is occluded by *any* other part, and 1 otherwise. If a part is partially occluded, only a subset of these binary variables are zeros. Letting $Z = \{z_1, z_2, ..., z_K\}$, the local likelihood term (3.2) can be rewritten as:

$$\begin{aligned} P(I|L,Z,\Theta) &= \prod_i P(I|l_i, z_i, \beta_i) && (3.10) \\ &\propto \prod_i \prod_{u\in\Omega(l_i)} \left(\frac{P_{l_i(u)}(f(I_u))}{P_{bg(u)}(f(I_u))}\right)^{z_i(u)} && (3.11) \\ &= \prod_i \prod_{u\in\Omega(l_i)} \left(\exp\left(\beta_{i(u)} f(I_u)\right)\right)^{z_i(u)} && (3.12) \end{aligned}$$

It is important to note that if all the occlusion variables z_i are consistent, the global likelihood $P(I|L,Z,\Theta)$ truly factorizes as (3.12). Similar to [99], we enforce the consistency of the occlusion variables using the following function:

$$\eta(l_j, z_{i(u)}; l_i) = \begin{cases} 0 & \text{if } l_j \text{ occludes } l_i,\ u \in \Omega(x_j), \text{ and } z_{i(u)} = 1 \\ 1 & \text{otherwise} \end{cases}$$

The consistency relationship of occlusion variable z_i and z_j can be enforced by the following potential function:

$$\psi^O(l_i, z_i, x_j, z_j) = \prod_{u\in\Upsilon} \eta(x_j, z_{i(u)}; x_i)\eta(x_i, z_{j(u)}; x_j) \quad (3.13)$$

Letting $\mathcal{E}_O$ be the set of edges corresponding to pairs of parts that are prone to occlusions, and defining $P_O(L,Z) \propto \prod_{(i,j)\in\mathcal{E}_O} \psi^O_{i,j}(l_i, z_i, l_j, z_j)$, we obtain the final occlusion sensitive version of our model:

$$P(L|I, Z, \Theta) \propto P(L|\alpha)P_O(L, Z)P(I|L, Z, \beta) \tag{3.14}$$

Occlusion-sensitive message passing: Now we discuss how to do message passing that involves occlusion variables z_i. Similar to previous work [99, 94], we assume that potentially occluding parts have a known relative depth in order to simplify the formulation. In general, one could introduce another discrete hidden variable indicating the relative depth order between parts and perform inference for each value.

Our inference scheme is similar to [99]. It is based on the following intuition. Suppose part j is occluding part i and we have a distribution of $P(l_j)$, we can use $P(l_j)$ to calculate an occlusion probability $P[z_{i(u)} = 0]$ for each pixel u. Then we can discount the image evidence at pixel u according to $P[z_{i(u)} = 0]$ when we use that image evidence to infer the configuration of l_i. If $P[z_{i(u)} = 0]$ is close to 1, it means pixel u has a higher probability of being claimed by part j. In this case, we will discount more of the image evidence at u. In the extreme case of $P[z_{i(u)} = 0]$ approaches 0 for all $\{u : u \in \Upsilon\}$, it is equivalent to inference without occlusion reasoning.

Consider the BP message sent from l_j to (l_i, z_i) in message passing. At this point, we already have a pseudo-marginal $P(\hat{l_j}|I))$ (it is the true marginal $P(l_j|I)$ if the underlying graph structure is a tree, and the message is passed from the root to the leaves). If l_i lies in front of l_j (remember that we known the depth order), the BP message $\mu_{j,i(u)}(z_{i(u)})$ is uninformative. If l_i is occluded and l_j is the only potentially occluding part, we firstly determine an approximation to the marginal occlusion probability $\nu_{i(u)} \approx Pr[z_{i(u)} = 0]$. If we think of $P(\hat{l_j}|I)$ as a 3D image (x, y, θ), $\nu_{i(u)}$ (which can be thought as a 2D image) can be efficiently calculated by convolving $P(\hat{l_j}|I)$ with rotated version of a uniform rectangle (with size proportional to the size of l_j) filter, then summing over θ dimension. Then the BP approximation to l_i can be written in terms of these marginal occlusion probabilities (see [99] for the rationale behind (3.15)):

$$\begin{aligned}
P(I|l_i) &\propto \prod_{u \in \Omega(l_i)} \left[\nu_{i(u)} + (1 - \nu_{i(u)}) \left(\frac{P_{l_i(u)}(f(I_u))}{P_{bg(u)}(f(I_u))} \right) \right] & (3.15) \\
&= \prod_{u \in \Omega(l_i)} \left[\nu_{i(u)} + (1 - \nu_{i(u)}) \exp\left(\beta_{i(u)} f(I_u) \right) \right] & (3.16) \\
&\approx \prod_{u \in \Omega(l_i)} \left[\exp\left(\left(1 - \nu_{i(u)}\right) \beta_{i(u)} f(I_u) \right) \right] & (3.17) \\
&= \exp\left(\sum_{u \in \Omega(l_i)} \left(\left(1 - \nu_{i(u)}\right) \beta_{i(u)} f\left(I_u\right) \right) \right) & (3.18) \\
&= \exp\left(\beta_i g_i(I(l_i), \nu_i) \right) & (3.19)
\end{aligned}$$

where $g_i(I(l_i), \nu_i)$ is a function similar to $f_i(I(l_i))$, but instead of returning 1, it returns a fractional number $(1 - \nu_{i(u)})$ at pixel u if I_u is an edge point. The approximation in (3.17) is based on the fact that absolute values of $\beta_{i(u)}$ are usually small (e.g., less than 0.6 in our experiments). When $|x|$ is small, $\exp(x)$ can be approximated by $1 + x$ based on the truncated Taylor expansion of $\exp(x)$.

Unlike previous methods [99, 94] which handle occlusion reasoning using sampling, our final result (3.19) has a surprisingly simple form. It can be efficiently calculated by first getting $g_i(I(l_i), \nu_i)$ through a simple dot-product between $f(I)$ (a binary 2D edge map of the whole image I) and $(1 - \nu_i)$ (a 2D image of occlusion marginals), then convolving g_i with rotated versions of β_i. The dot-product has the nice intuition of discounting the image evidences by their occlusion variables. Our method can be applied efficiently and exhaustively over all the image pixel locations. This is due to the convolution trick. However, if the structure of the graphical model is not a tree, one has to use loopy belief propagations. In that case, the convolution trick is no longer valid, since the message stored at a node is no longer in a simple form that allows the derivation of (3.19) to go through. This further justifies the advantage of using tree-structured models.

3.4 Experiments

CMU MoBo dataset: We first test our algorithm on the rescaled versions of side-view persons of CMU mobo dataset [36] for the occlusion reasoning. Since people's right arm in this dataset is almost always occluded, we only try to infer one arm. We use the background

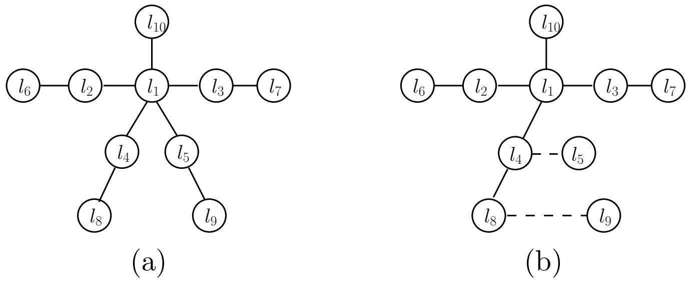


Figure 3.4: Two tree structures used on CMU Mobo dataset. We use dashed lines to indicate occlusion relationships, rather than spatial constraints.

subtraction masks that come with this dataset to remove the edges found in the background.

We use the two tree structures shown in Fig. 3.4. The first tree captures the kinematic spatial constraint. The second tree captures the occlusion relationships between the left and right legs. Inference in the second tree uses the message passing algorithm described in Sect. 3.3. Learning the model parameters is a bit tricky. If we use CMU mobo dataset for training, we will probably end up with a strong spatial prior specifically tuned to side-view walking. Instead, we learn the model parameters $\Theta = \{\alpha_i, \beta_i\}$ using the same training set in our second experiment (see below). That dataset contains images of people with a variety of poses. We manually set the weights of these two trees to be equal, since we do not have appropriate datasets with ground truths, and we do not want to learn the parameters from the mobo dataset. In principle, this parameter can be learned from some labeled dataset where the relative depth order of parts is known.

Some of the sample results are shown in Fig. 3.5. We can see that the single tree model tends to put the legs on top of each other. But our method correctly infers the configurations of both legs. To quantify the results, we manually label 300 mobo images as ground truths and measure their *perplexity* (or *negative log-probability* [78]) under the learned model. Instead of measuring the perplexity for the whole body pose L, we measure them separately for each body part $l_i (i = 1, 2, ..., K)$ to emphasize the effect of occlusion reasoning between two legs. As shown in Table 3.1, our method achieves lower perplexity on the lower and upper right legs. The perplexities for other body parts are not shown in the table since they are the same for both methods. This is because we have only modeled the occlusion relationships between the legs.

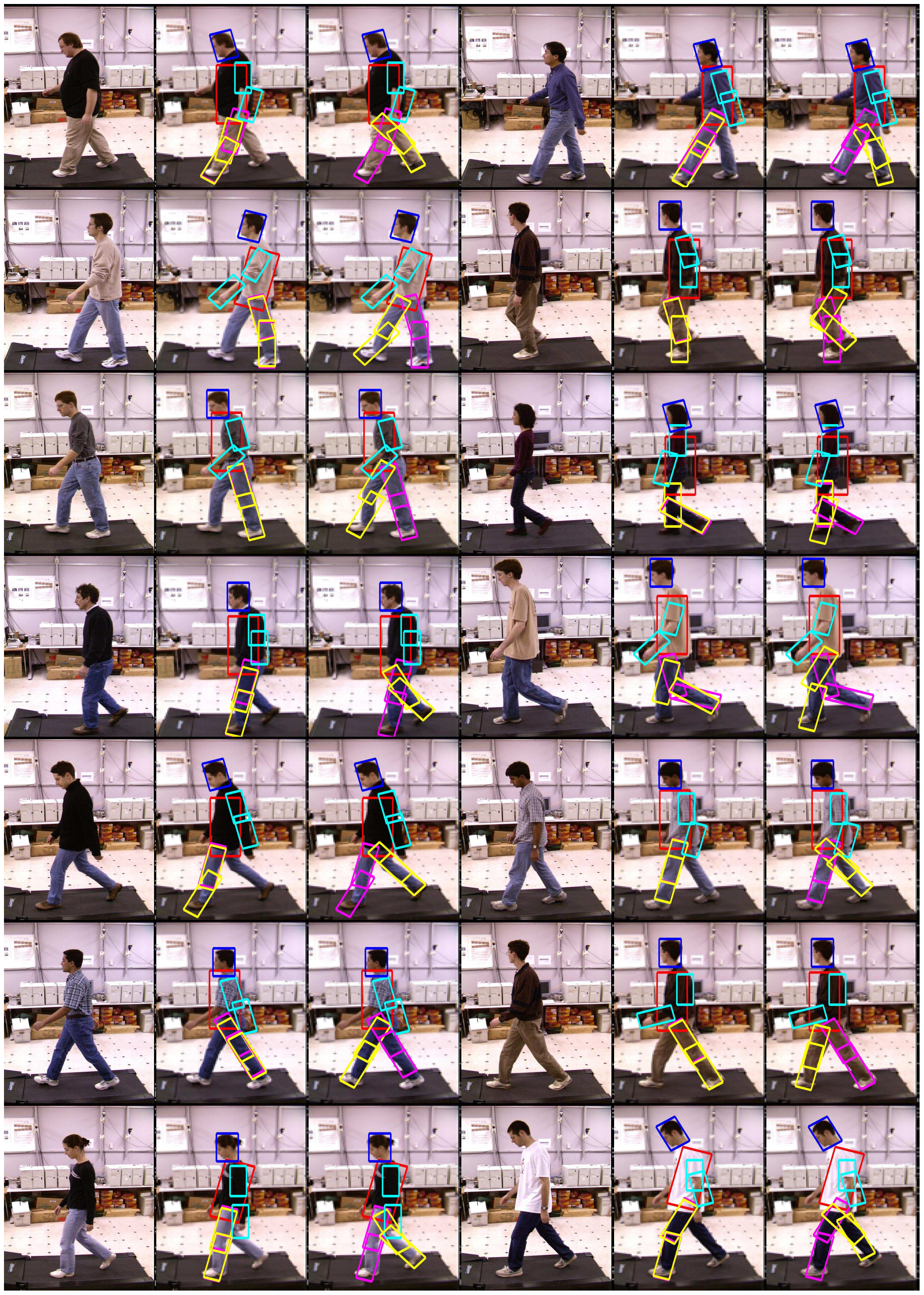

(a) (b) (c) (a) (b) (c)

Figure 3.5: Sample results on the CMU mobo dataset: (a) original images; (b) results of using one kinematic tree; (c) results of using multiple trees for occlusion reasoning.

Part	Perplexity(two trees)	Perplexity(one tree)
ru-leg	32.4939	33.9706
rl-leg	26.7597	33.6693

Table 3.1: Quantitative measurement on mobo dataset for the right upper and lower legs. Smaller perplexities mean better performance.

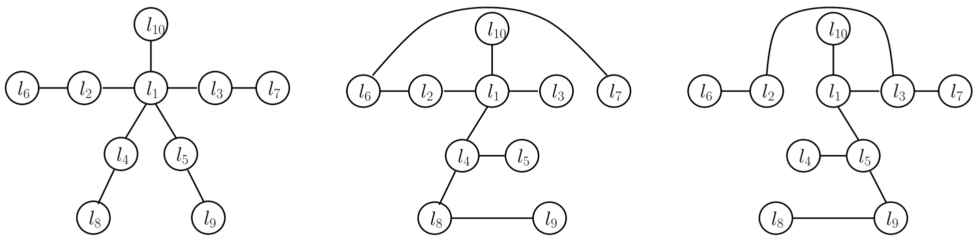


Figure 3.6: Three tree structures used on the people dataset.

People dataset: We test our algorithm on the people dataset used in previous work [78]. This dataset contains 305 images of people in various poses. First 100 images and their mirror-flipped versions are used for training, and the remaining 205 images for testing. We manually select three tree structures shown in Fig. 3.6, although it will be an interesting future work on how to automatically learn the tree structure at each iteration in an efficient way. We visualize the distribution $P(L|I)$ as a 2D image using the same technique in [78], where the torso is rendered as red, the upper-limbs as green, the lower-limbs and the head as blue. Some of the parsing results are shown in Fig. 3.7. We can see that our parsing results are much clearer than the ones using the kinematic tree. In many images, the body parts are almost clearly visible from our parsing results. In the results of using the kinematic tree, there are many white pixels, indicating high uncertainty about body parts at those locations. But with multiple trees, a lot of these white pixels are cleaned up. It is plausible that if we sample the part candidates l_i according to $P(l_i|I)$ and use them as the inputs to other pose estimation algorithms (e.g., Ren et al. [84]), the samples generated from our parsing results are more likely to be the true part locations.

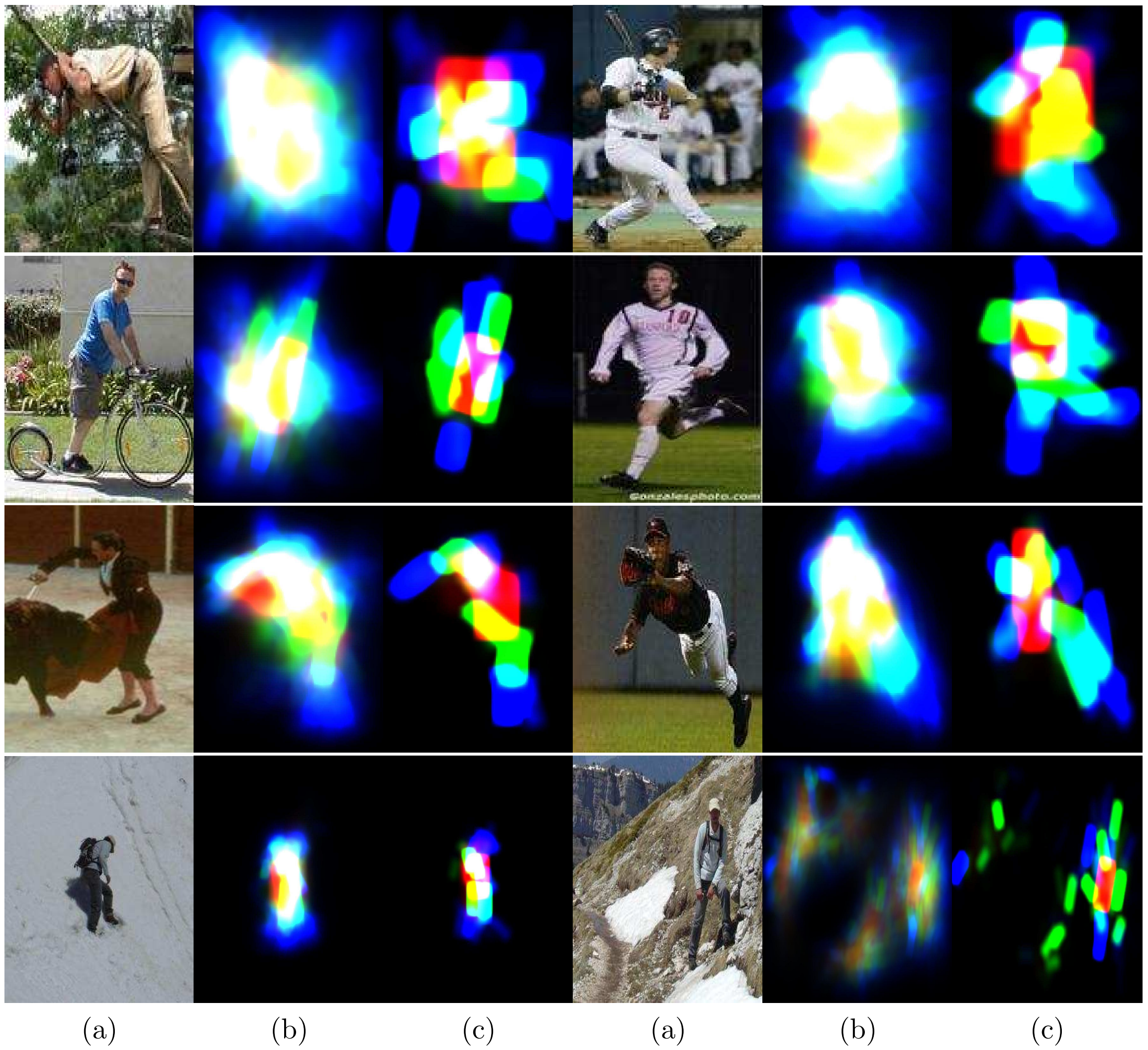


Figure 3.7: Sample results on the people dataset: (a) original images; (b) results of using one kinematic tree; (c) results of using multiple trees.

3.5 Summary

We have presented a framework for modeling human figures as a collection of tree-structured models. This framework has the computational advantages of previous tree-structured models used for human pose estimation. At the same time, it models a richer set of constraints between body parts. We demonstrate our results on side-walking persons in CMU mobo dataset, and a challenging people dataset with substantial pose variations.

Human pose estimation is an extremely difficult computer vision problem. The solution of this problem probably requires the symbiosis of various kinds of visual cues. Our framework provides a flexible way of modeling dependencies between non-connected body parts.

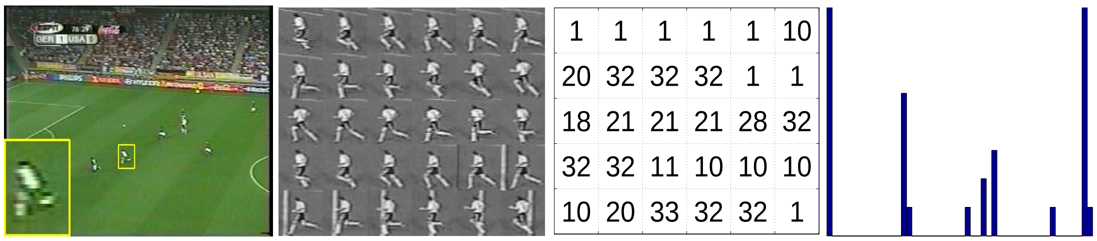
Chapter 4

Topic Models for Action Recognition

Various visual cues (e.g., motion [21, 24, 59, 74] and shape [100]) can be used for recognizing actions. In this chapter, we focus on recognizing the action of a person in an image sequence based on motion cues. We develop two novel models for human action recognition based on the “bag-of-words” paradigm.

Our models are motivated by the recent success of “bag-of-words” representations for object recognition problems in computer vision. The common paradigm of these approaches consists of extracting local features from a collection of images, constructing a codebook of visual words by vector quantization, and building a probabilistic model to represent the collection of visual words. While these models of an object as a collection of local patches are certainly not “correct” ones, for example, they only model a few parts of objects and often ignore many structures, they have been demonstrated to be quite effective in object recognition tasks [27, 35, 56].

In this chapter we explore the use of two similar models for recognizing human actions. Figure 4.1 shows an overview of our “bag-of-words” representation. In our models, each frame of an image sequence is assigned to a visual word by analyzing the motion of the person it contains. The unordered set of these words over the image sequence becomes our bag of words. As with the object recognition approaches, some structures have been lost by moving to this representation. However, this representation is much simpler than one that explicitly models temporal structures. Instead we capture “temporal smoothing”



1	1	1	1	1	10
20	32	32	32	1	1
18	21	21	21	28	32
32	32	11	10	10	10
10	20	33	32	32	1

Figure 4.1: The processing pipeline of the "bag-of-words" representation: (a) given a video sequence, (b) track and stabilize each human figure, (c) represent each frame by a "motion word", (d) ignore the ordering of words and represent the image sequences of a tracked person as a histogram over "motion words".

via co-occurrence statistics amongst these visual words, i.e., which actions tend to appear together in a single track. For example, in a single track of a person, the combination of "walk left" and "walk right" actions is much more common than the combination of "run left" "run right" "run up" "run down". We note that there has been previous work (e.g., Yamato et al. [119], Bobick & Wilson [14], Xiang & Gong [118]) that tries to model the full dynamics of videos using sophisticated probabilistic models (e.g., hidden Markov models, dynamic Bayesian networks). The problem with this approach is that those sophisticated models impose too many assumptions and constraints (e.g., the independence assumption of hidden Markov models) in order to be tractable. It is also hard to learn those models since there are usually a large number of parameters that need to be set. Instead, our methods can be considered as a way of imposing a "rough" constraint on the overall temporal structures of videos, without worrying about the detailed temporal information between adjacent frames. In this chapter, we provide evidence that this simple representation can be quite effective in recognizing actions.

Our models are based on the Latent Dirichlet allocation (LDA) model [12] and the Correlated Topic Model (CTM) [10]. Topic models, such as, LDA, CTM, probabilistic Latent Semantic Analysis (pLSA) [38], and their variants, have been applied to various computer vision applications, such as scene recognition [15, 28], object recognition [32, 87, 95], action recognition [70], human detection [8], etc.

Despite the great success achieved, there are some unsolved, important issues remaining in this line of research. First of all, most of the previous approaches use their models for some specific recognition problem, say object class recognition. However, there is no guarantee that the latent topics found by their algorithms will necessarily correspond to object classes. Secondly, the features used in these approaches are usually SIFT-like local features computed at locations found by interest-point detectors. The only exceptions are histogram of oriented gradients in Bissacco et al. [8] and multiple segmentations in Russell et al. [87]. Features based on local patches may be appropriate for certain recognition problems, such as scene recognition or object recognition. But for human action recognition, it is not clear that they can be sufficiently informative about the action being performed. Instead, we use descriptors that can capture the large-scale properties of human figures, and compare these results to approaches using local patches.

In this chapter, we attempt to address the above mentioned issues in two aspects. First of all, we introduce a new "bag-of-words" representation for image sequences. Our representation is dramatically different from previous ones (e.g., Niebles et al. [70]) in that we represent a frame in an image sequence as a "single word", rather than a "collection of words" computed at some spatial-temporal interest points. Our main motivation for this new representation is that human actions may be better characterized by large-scale features, rather than local patches. For example, consider the image in Fig. 4.2, it is very easy to see that the large scale motion descriptors (described in Section 4.1.1) capture some important characteristics of this motion, e.g., the movements of the legs. It is not obvious that one can recognize this action by just looking at several small patches in the image. Secondly, we propose two new topic models called *Semi-Latent Dirichlet Allocation (S-LDA)* and *Semi-latent Correlated Topic Model (S-CTM)*, respectively. The major difference between our models and the traditional latent topic model (e.g, *LDA* and *CTM*) is that some of the latent variables in LDA and CTM are observed during the training stage in S-LDA and S-CTM. We show that by naturally pushing the information provided by class labels of training data directly into our model, we can guide the previously latent topics to be our class labels, and consequently achieve much better performance. We notice that the idea of adding supervision to the LDA model has been applied in various ways in other work, e.g., supervised topic models [11], labeled LDA [34].

There are other alternative methods to train an action recognition system. For example, one can train a discriminative classifer (e.g., SVM) to recognize the action of each frame individually. But this approach ignores the contextual information provided by different frames in a video. We will demonstrate experimentally that our approach performs better than this alternative.

The contributions the work presented in this chapter are three-fold. First, we propose a novel bag-of-words representation for video sequences. Second, we introduce two semi-latent topic models in which class labels of the frames in a video are naturally exploited in the learning process. Third, we present extensive experimental results to show that the proposed models achieve state-of-the-art recognition accuracies on a large variety of datasets.

4.1 Our Approach

Our approach follows the bag-of-words framework. But our models are different from previous bag-of-words models (e.g., Niebles et al. [70]) in two major aspects. First of all, our method represents a frame as a single word, rather than a collection of words from vector quantization of space-time interest points. In other words, a "word" corresponds to a "frame", and a "document" corresponds to a "video sequence" in our representation. Secondly, our model is trained in a supervised fashion. We will show that by utilizing the class labels, we can greatly simplify the training algorithm, and achieve much better recognition accuracy.

4.1.1 Motion Features and Codebook

We use the motion descriptor in Efros et al. [24] to represent the video frames. This motion descriptor has been shown to perform reliably with noisy image sequences, and has been applied in various tasks, such as action classification, motion synthesis, etc.

To calculate the motion descriptor, we first need to track and stabilize the persons in a video sequence. We use an automatic human detection method in some of our experiments. But any tracking or human detection methods can be used, since the motion descriptor we use is very robust to jitters introduced by the tracking.

Given a stabilized video sequence in which the person of interest appears in the center of the field of view, we compute the optical flow at each frame using the Lucas-Kanade [62] algorithm. The optical flow vector field F is then split into two scalar fields F_x and F_y, corresponding to the x and y components of F. F_x and F_y are further half-wave rectified into four non-negative channels F_x^+, F_x^-, F_y^+, F_y^-, so that $F_x = F_x^+ - F_x^-$ and $F_y = F_y^+ - F_y^-$. These four non-negative channels are then blurred with a Gaussian kernel and normalized to obtain the final four channels Fb_x^+,Fb_x^-,Fb_y^+,Fb_y^- (see Fig. 4.2).

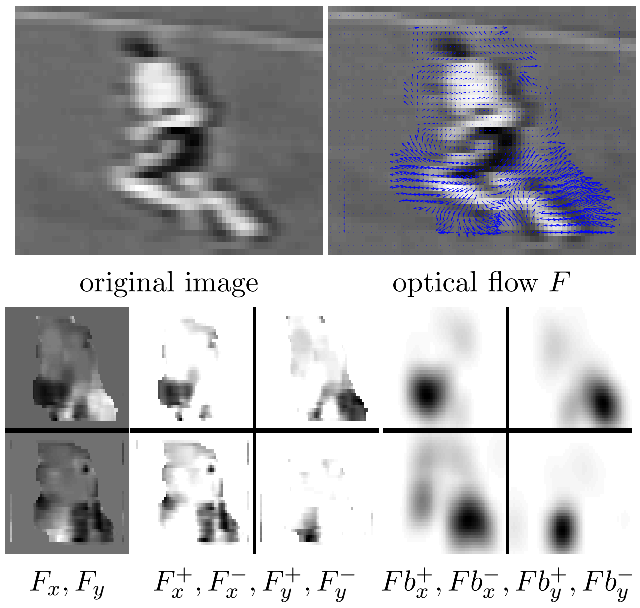

original image
optical flow F
F_x, F_y
$F_x^+, F_x^-, F_y^+, F_y^-$
$Fb_x^+, Fb_x^-, Fb_y^+, Fb_y^-$

Figure 4.2: Construction of the motion descriptor. See Section 4.1.1 for details.

The motion descriptors of two different frames are compared using a version of the normalized correlation. Suppose the four channels for frame i of sequence A are a_1, a_2, a_3 and a_4, similarly, the four channels for frame j of sequence B are b_1, b_2, b_3 and b_4, then the similarity between frame A_i and frame B_j is:

$$S(A_i, B_j) = \sum_{t \in T} \sum_{c=1}^{4} \sum_{x,y \in I} a_c^{i+t}(x,y) b_c^{j+t}(x,y) \tag{4.1}$$

where T and I is the temporal and spatial extent of the motion descriptors. We choose $T = 10$ in all of our experiments. The final dimensionality of the feature vector is $4 \times T \times I$.

To construct the codebook, we randomly select a subset from all the frames, compute the affinity matrix on this subset of frames, where each entry in the affinity matrix is the similarity between frame i and frame j calculated using the normalized correlation described above. Then we run k-medoid clustering on this affinity matrix to obtain V clusters. Codewords are then defined as the centers of the obtained clusters. In the end, each video sequence is converted to the "bag-of-words" representation by replacing each frame with its corresponding codeword and removing the temporal information.

4.1.2 Latent Topic Models

Our models for video sequences are based on latent topics models, in particular, the Latent Dirichlet Allocation (LDA) [12] model and the Correlated Topic Model (CTM)[10]. In the following, we briefly introduce both of them using the terminology in our context.

Latent Dirichlet Allocation

Suppose we are given a collection D of video sequences $\{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$. Each video sequence $\mathbf{w}$ is a collection of frames $\mathbf{w} = (w_1, w_2, ..., w_N)$, where w_i is the motion word representing the i-th frame. A motion word is the basic item from a codebook (see Section 4.1.1) indexed by $\{1, 2, ..., V\}$.

The LDA model assumes there are K underlying action labels (i.e., topics) according to which video sequences are generated. For example, a typical video sequence can be composed of frames with labels "walking left", "running left", 'standing still', etc. Each action is represented by a multinomial distribution over the V motion words. The "motion words" can be thought of as a set of "prototypes" obtained by clustering all the frames in the training set. See Fig. 4.4 for an illustration. A video sequence is generated by sampling a mixture of these actions, then sampling motion words conditioning on a particular action. The generative process of LDA for a video sequence $\mathbf{w}$ in the collection can be formalized as follows (see Fig. 4.3(a)):

1. Choose $\theta \sim \text{Dir}(\alpha)$
2. For each of the N motion words w_n:
   (a) Choose an action label (i.e., topic) $z_n \sim \text{Mult}(\theta)$;
   (b) Choose a motion word w_n from $w_n \sim p(w_n|z_n, \beta)$, a multinomial probability conditioned on z_n.

The parameter θ indicates the mixing proportion of different action labels in a particular video sequence. α is the parameter of a Dirichlet distribution that controls how the mixing proportion θ varies among different video sequences. β is the parameter of a set of multinomial distributions, each of them indicates the distribution of motion words within a

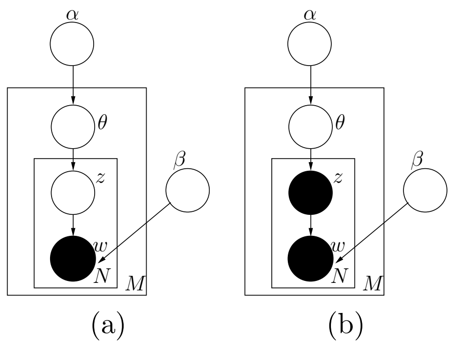


Figure 4.3: Graphical representation of: (a) LDA model, adopted from Blei et al. [12]; (b) S-LDA for training. Note the difference from LDA is that z is observed in this case.

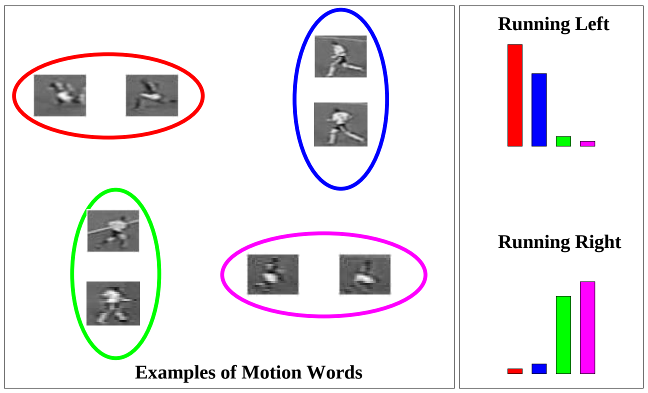


Figure 4.4: Illustrations of topic models. Left: examples of clustering frames into "motion words", where each cluster (denoted by a color ellipse) corresponds to a "word". The red and blue clusters roughly correspond to two types of "running left", while the green and pink clusters roughly correspond to "running right". Right: examples of multinomial distributions (i.e., the parameter β) on words for two difference action labels. For the "running left" action, the multinomial distribution has large components on the red and blue components, while the distribution for "running right" has large components on the green and pink components.

particular action label. The probability of a video $\mathbf{w} = \{w_1, w_2, ..., w_n\}$ is:

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta$$

Given a collection of video sequences $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$, learning a LDA model involves finding the model parameters α and β that maximize the log likelihood of the data $\ell(\alpha, \beta) = \sum_{d=1}^{M} \log P(\mathbf{w}_d|\alpha, \beta)$. This parameter estimation problem can be solved by the variational EM algorithm developed in Blei et al. [12].

Correlated Topic Model

Blei & Lafferty [10] point out that the Dirichlet prior on topic proportions $\theta \sim \text{Dir}(\alpha)$ in LDA does not properly model the correlation of different topics in the documents. To address this limitation, they propose a new topic model called the Correlated Topic Model (CTM). CTM uses the logistic normal distribution, rather than the Dirichlet distribution, as the prior distribution of the topic proportions. The generative process of CTM is as follows:

1. Choose $\eta \sim \mathcal{N}(\mu, \Sigma)$
2. For each of the N motion words w_n:
   (a) Choose an action label (i.e., topic) $z_n \sim \text{Mult}(\theta)$, where $\theta_i = \exp \eta_i / \sum_j \exp \eta_j$;
   (b) Choose a motion word w_n from $w_n \sim \text{Mult}(\beta_{z_n})$

This generative process is identical to that of LDA except that the topic proportions are now drawn from a logistic normal rather than a Dirichlet distribution. The parameter estimation problem in CTM can be solved by the variational EM algorithm [10].

4.1.3 Semi-Latent Topic Models for Training

In the original LDA or CTM, we are only given the words $(w_1, w_2, ..., w_N)$ in each video sequence, but we do not know the topic z_i for the word w_i, nor the mixing proportion θ of topics in the sequence. In order to use LDA for classification problems, people have applied various tricks. For example, Blei et al. [12] use LDA to project a document onto the topic simplex, then train an SVM model based on this new representation, rather than the original vector representation of a document based on words. Although this simplex

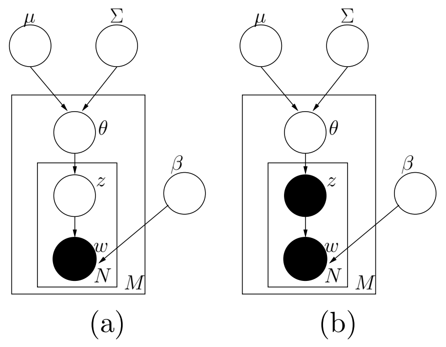


Figure 4.5: Graphical representation of: (a) CTM model, adopted from Blei et al. [10]; (b) S-CTM for training. Note the difference from CTM is that z is observed in this case.

is a more compact representation for the document, the final SVM classifier based on this new representation actually performs worse than the SVM classifier trained on the original vector representation based on words. Sivic et al. [95] use a simpler method by classifying an image to a topic in which the latent topics of this image is most likely to be drawn from. There are two problems with this approach. First of all, there is no guarantee that a "topic" found by LDA or CTM corresponds to a particular "object class". Secondly, it is not clear how many "topics" to choose.

Semi-Latent Dirichlet Allocation (S-LDA)

We are interested in the action classification problem, where all the frames in the training video sequences have action class labels associated with them. In this case, there is no reason to ignore this important information. In this section, we introduce a modified version the LDA model called *Semi-Latent Dirichlet Allocation (S-LDA)*. S-LDA utilizes class labels by enforcing a one-to-one correspondence between topics and class labels. Since we use a word w_i to represent a frame in a video sequence $\mathbf{w} = (w_1, w_2, ..., w_N)$, the topic z_i for the word w_i is simply the class label of w_i. The graphical representation of S-LDA model is shown in Fig. 4.3(b). We should emphasize that the model in Fig. 4.3(b) is only for training (i.e., estimating α and β). In testing, we will use the same model shown in Fig. 4.3(a), together with estimated model parameters α and β.

Our model has several major advantages over previous approaches of using a topic model for classification problems. First of all, the training process of the S-LDA model is much easier than the original LDA. Secondly, we can achieve much better recognition accuracy by taking advantage of the class labels (see Section 4.2). In addition, we do not have to choose the number of latent topics in S-LDA, since it is simply the number of class labels.

In LDA (see Fig. 4.3(a)), the parameters α and β are coupled, conditioning on the observed words $\mathbf{w}$. In that case, the model parameters (α and β) have to be estimated jointly, which is difficult. Various approximation approaches (e.g., sampling, variational EM, etc) have to be used. However, in S-LDA (Fig. 4.3(b)), the parameters α and β become independent, conditioning on observed words $\mathbf{w}$ and their corresponding topics (i.e., class labels) $\mathbf{z}$. So we can estimate α and β separately, which makes the training procedure much easier. In the following, we describe the details of how to estimate these parameters.

The parameter β can be represented by a matrix of size $K \times V$, where K is the number of possible topics (i.e., class labels) and V is the number of possible words. The i-th row of this matrix (β_i) is a V-dimensional vector that sums to 1. β_i is the parameter of a multinomial distribution, which defines the probability of drawing each word in the i-th topic. The maximum-likelihood estimate of β_i can be calculated by simply counting the frequency of each word appearing together with topic z_i, i.e., $\beta_{ij} = n_{ij}/n_{i\cdot}$, where $n_{i\cdot}$ is the count of the i-th topic in the corpus, and n_{ij} is the count of i-th topic with j-th word in the corpus.

The Dirichlet parameter α can be estimated from a "*Dirichlet-multinomial*" distribution (or *Polya* distribution), which is a compound distribution where θ is drawn from a Dirichlet and then a sample of discrete outcomes $\mathbf{z}$ is drawn from a multinomial with probability vector θ. The resulting distribution over $\mathbf{z}$ is $p(\mathbf{z}|\alpha) = \int_\theta p(\mathbf{z}|\theta)p(\theta|\alpha)d\theta$. Given a set of $\{\mathbf{z_1}, \mathbf{z_2}, ..., \mathbf{z_M}\}$, the maximum-likelihood estimate of α can be calculated using Newton-Raphson iterations [64].

Semi-latent Correlated Topic Model (S-CTM)

We can modify the CTM model in a similar way to obtain the *Semi-latent Correlated Topic Model (S-CTM)* for training. The graphical representation of S-CTM is shown in Fig. 4.5(b).

Similarly, the training algorithm of S-CTM is much simpler than CTM, due to the fact that the parameters μ and Σ are decoupled from the parameters β. The maximum likelihood estimation of β is identical to that in S-LDA. Given a set $\{\mathbf{z_1}, \mathbf{z_2}, ..., \mathbf{z_M}\}$, the parameters μ and Σ can be estimated using the EM algorithm [40].

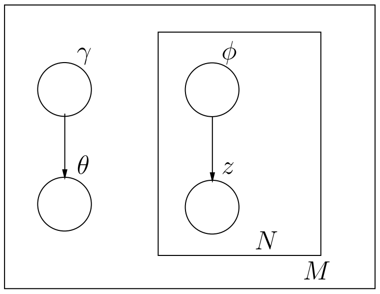


Figure 4.6: Graphical representation of the variational distribution for LDA

4.1.4 Classification of New Video Sequences

Given a new video sequence for testing, we would like to classify each frame in the sequence. Suppose the testing video sequence is represented as $\mathbf{w} = (w_1, w_2, ..., w_N)$, i.e., there are N frames in the sequence, and the n-th frame is represented by the motion word w_n. Then, we need to calculate $p(z_n|\mathbf{w})$ $(n = 1, 2, ..., N)$. The frame w_n is classified to be action class k if $k = \arg\max_j p(z_n = j|\mathbf{w}, \alpha, \beta)$. Notice that we use $p(z_n|\mathbf{w})$ instead of $p(z_n|w_n)$ for classification. This reflects our assumption that the class label z_n not only depends on its corresponding word w_n, but also depends on the video sequence $\mathbf{w} = (w_1, w_2, ..., w_N)$ as a whole.

Inference of S-LDA

To calculate $p(z_n|\mathbf{w}, \alpha, \beta)$ in S-LDA, we use the variational inference algorithm proposed in Blei et al. [12]. The basic idea of the variational inference is to approximate the distribution $p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$ by a simplified family of variational probability distributions $q(\theta, \mathbf{z})$ with the form $q(\theta, \mathbf{z}) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n)$. The graphical representation of $q(\theta, \mathbf{z}|\gamma, \phi)$ is shown in Fig. 4.6.

In order to make the approximation as close to the original distribution as possible, we need to find (γ^*, ϕ^*) that minimize the Kullback-Leibler (KL) divergence between the variational distribution $q(\theta, \mathbf{z}|\gamma, \phi)$ and the true distribution $p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$, i.e., $(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} D(q(\theta, \mathbf{z}|\gamma, \phi) \| p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta))$, where $D(\cdot|\cdot)$ is the KL divergence. Finding (γ^*, ϕ^*) can be achieved by iteratively updating (γ, ϕ) using the following update rules (see Blei et al. [12] for detailed derivations):

$$\begin{aligned}\hat{\phi}_{ni} &\propto \beta_{iv}\exp(\Psi(\gamma_i)-\Psi(\sum_{j=1}^{K}\gamma_j)) &(4.2)\\ \hat{\gamma}_i &= \alpha_i+\sum_{n=1}^{N}\phi_{ni} &(4.3)\end{aligned}$$

Several insights can be drawn from examining the variational parameters $(\gamma^*(\mathbf{w}),\phi^*(\mathbf{w}))$. First of all, $(\gamma^*(\mathbf{w}),\phi^*(\mathbf{w}))$ are video-specific. Also notice that $\mathrm{Dir}(\gamma^*(\mathbf{w}))$ is the distribution from which the mixing proportion θ for the video sequence $\mathbf{w}$ is drawn. We can imagine that if we draw a sample $\theta\sim\mathrm{Dir}(\gamma^*(\mathbf{w}))$, θ will tend to peak towards the true mixing proportion θ^* of actions for the video $\mathbf{w}$. So the true mixing proportion θ^* can be approximated by the empirical mean of a set of samples $\theta^{(i)}$ drawn from $\mathrm{Dir}(\gamma^*(\mathbf{w}))$. The second insight comes from examining the ϕ_n parameters. These distributions approximate $p(z_n|w_n)$. The third insight is that, since the topic z_n is drawn from $\mathrm{Mult}(\theta^*)$, θ^* is an approximation of $p(z_n)$. Then we can get $p(z_n|\mathbf{w})\propto p(z_n|\theta^*)p(z_n|w_n)\approx\theta^*_{z_n}\phi_{z_nw_n}$.This equation has a very appealing intuition. It basically says the class label z_n is determined by two factors, the first factor $\theta^*_{z_n}$ tells us the probability of generating topic z_n in a document with mixing proportion θ^*, the second factor $\phi_{z_nw_n}$ tells us the probability of generating topic z_n conditioning on a particular word w_n. We can classify frame n according to $p(z_n|\mathbf{w})$. If we know the video $\mathbf{w}$ only contains one action, the classification label for the entire video can be obtained by majority voting.

Inference of S-CTM

For S-CTM, the probability $p(z_n|\mathbf{w},\mu,\Sigma)$ can be similarly calculated using a variational inference algorithm [10]. For fixed model parameters $\{\beta,\mu,\Sigma\}$, the log probability of a new document $\mathbf{w}=\{w_1,...,w_n\}$ can be bound using Jensen's inequality:

$$\begin{aligned}\ell = &\ \log p(w_{1:N}|\mu,\Sigma,\beta)\\ \geq &\ \mathbf{E}_q[\log p(\theta|\mu,\Sigma)]\\ + &\ \sum_{n=1}^{N}\mathbf{E}_q[\theta^\top z_n]+\sum_{n=1}^{N}\mathbf{E}_q[\log p(w_n|z_n,\beta)]+\mathbf{H}(q)\\ + &\ \sum_{n=1}^{N}\Big(-\zeta^{-1}\big(\sum_{i=1}^{K}\mathbf{E}_q[\exp\{\theta_i\}]\big)+1-\log(\zeta)\Big) \qquad (4.4)\end{aligned}$$

where $\mathbf{H}(\cdot)$ is the entropy of a distribution, $\mathbf{E}_q(\cdot)$ is the expectation with respect to a

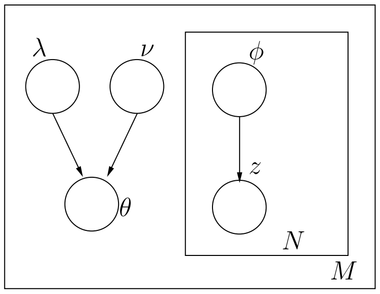


Figure 4.7: Graphical representation of the variational distribution for CTM

variational distribution $q(\theta, \mathbf{z})$. The variational probability distribution $q(\theta, \mathbf{z})$ is chosen to have the form $q(\theta, \mathbf{z}) = \prod_{i=1}^{K} q(\theta_i|\lambda_i, \nu_i^2) \prod_{n=1}^{N} q(z_n|\phi_n)$, where $q(z_n|\phi_n)$ is a multinomial distribution, and $q(\theta_i|\lambda_i, \nu_i^2)$ is a univariate Gaussian distribution with mean λ_i and variance ν_i^2. The graphical representation of the variational distribution is shown in Fig. 4.7.

For fixed model parameters $\{\beta, \mu, \Sigma\}$, the optimal values for the variational parameters $\{\lambda^*, \nu^*, \zeta^*\}$ can be found using coordinate ascent, repeatedly optimizing (4.4) with respect to each parameter while holding others fixed.

Maximizing (4.4) with respect to ζ and ϕ_n gives the following update rules:

$$\hat{\zeta} = \sum_{i=1}^{K} \exp\{\lambda_i + \nu_i^2/2\} \tag{4.5}$$

$$\hat{\phi}_{n,i} \propto \exp\{\lambda_i\}\beta_{i,w_n}, \qquad i \in \{1, ..., K\} \tag{4.6}$$

Maximizing (4.4) with respect to λ_i and ν_i^2 does not yield analytic solutions, but conjugate gradient algorithms can be used with derivatives (see [10] for detailed derivations):

$$\frac{d\ell}{\lambda} = -\Sigma^{-1}(\lambda - \mu) + \sum_{n=1}^{N} \phi_{n,1:K} - (N/\zeta)\exp\{\lambda + \nu^2/2\}$$

$$\frac{d\ell}{d\nu_i^2} = -\Sigma_{ii}^{-1}/2 - N/2\zeta \exp\{\lambda + \nu_i^2/2\} + 1/(2\nu_i^2)$$

After obtaining $(\gamma^*(\mathbf{w}), \phi^*(\mathbf{w}))$ for the video $\mathbf{w}$, we can compute $p(z_n|\mathbf{w})$ using the same technique used in S-LDA, i.e. $p(z_n|\mathbf{w}) \propto p(z_n|\theta^*)p(z_n|w_n) \approx \theta^*_{z_n}\phi_{z_n w_n}$, where θ^* is the empirical mean of a set of samples $\theta^{(i)}$ drawn from $\mathrm{Dir}(\gamma^*(\mathbf{w}))$. The frame n is classified according to $p(z_n|\mathbf{w})$. For single-action videos, the classification label of the entire video can obtained by majority voting. Of course, this is not the optimal way to obtain the per-video classification from the per-frame classification – one can build a much more sophisticated model for this purpose. But this work focuses on per-frame classification, so we prefer a simple and straightforward method to obtain the class label of the whole video. We will show experimentally that this simple method already gives very good results.

4.2 Experiments

We test our algorithm on five datasets: KTH human motion dataset [90], Weizmann human action dataset [9], hockey dataset [61], soccer dataset [24], and a new ballet dataset [26]. See Fig. 4.8 for sample frames from each dataset. Since the first three datasets (KTH, Weizmann, hockey) only contain single-action video sequences, and most of the previously published results are on per-video classification, we will focus on per-video classification (using majority voting described in the previous section) on these three datasets. The video sequences in the last two datasets (soccer, ballet) contain multiple actions in a video sequence, so we cannot do per-video classification. We will only report per-frame classification results on them. For each dataset, we perform "leave-one-out" cross-validation. For the first two datasets (KTH, Weizmann), we leave the videos of one person as test data each time. And for the last three datasets (hockey, soccer, ballet) we leave one video as test data each time.

On the KTH and Weizmann datasets, we also show experimental results on using SVM with large-scale features and compare with previous work that uses SVM with local patch features. Since the classification algorithm is identical in this case, this will allow us to compare large-scale features and local patch features directly.

Our approaches are efficient. Most of the computation is spent on the features and codewords. After the bag-of-words representation is obtained, learning the model usually takes less than one minute, and inference on a new video only takes a few seconds in our unoptimized MATLAB implementation combined with existing codes in MATLAB/C [12, 10, 64, 40].

KTH dataset: The KTH human motion dataset contains six types of human actions (walking, jogging, running, boxing, hand waving and hand clapping) performed several times by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. Representative frames of this dataset are shown in Fig. 4.8. We first run an automatic preprocessing step to track and stabilize the video sequences, so that all the figures appear in the center of the field of view. The confusion matrix for per-video classification of S-LDA on the KTH dataset using 1050 codewords is shown in Fig. 4.9(a). We can see that the algorithm correctly classifies most actions. Most of the mistakes the algorithm makes are confusions between "running" and "jogging" actions. This is intuitively reasonable, since "running" and "jogging" are similar actions. On all of our datasets, the confusion matrices (per-frame or per-video) of S-LDA and S-CTM have similar patterns, so we will only show one confusion matrix on each dataset.

We also test the effect of the codebook size on the overall accuracy for both per-frame and per-video classification. The result is shown in Fig. 4.9(b). The best accuracy is achieved with 1050 codewords for both S-LDA and S-CTM, but the results are relatively stable. We compare our results with previous approaches on the same dataset, as shown in Table 4.1. Performances on KTH and Weizmann (see below) datasets are saturating – state-of-the-art approaches achieve near-perfect results. Also we should note that different methods listed in Table 4.1 have all sorts of variations in their experiment setups, e.g., different splits of training/testing data, whether some preprocessing (e.g., tracking, background subtraction) is needed, with or without supervision, whether per-frame classification can be done, whether a method handles multiple action classes in a video, etc. The results of our methods are comparable to other state-of-the-art approaches,although we emphasize that this is not a precise comparison due to the variations in the experiment setups mentioned before.

The KTH dataset only contains single-action videos. If one only needs per-video classification, the simplest approach is to train a discriminative classifier based on the large-scale features. In order to see how the large-scale features would perform in this case, we build a histogram representation based on the visual words computed from the large-scale features for each video, then train an SVM classifer based on the histogram. This is similar to what has been done in object recognition [121]. Using a linear SVM with the trade-off parameter $C = 1000$, the best per-video classification performance is 83.31%, which is much better than a similar approach in Schuldt et al. [90] that uses SVM based on histograms of visual words obtained from local patches. We have tried other different values for the parameter C. The results are relatively stable (within $\sim 3\%$). This demonstrates that the large-scale features outperform local patch features. However, using the same large scale features, our approach (S-LDA/S-CTM) still outperforms SVMs.

Weizmann dataset: The Weizmann human action dataset contains 83 video sequences showing nine different people, each performing nine different actions. We track and stabilize the figures using the background subtraction masks that come with this dataset. Some

methods	**accuracy(%)**
S-LDA	91.20
S-CTM	90.33
Liu & Shah[60]	94.16
Schindler & Van Gool[88]	92.70
Wong et al. [117]	91.60
Jhuang et al. [45]	91.70
Nowozin et al. [71]	87.04
Niebles et al. [70]	81.50
Dollár et al. [23]	81.17
Schuldt et al. [90]	71.72
Ke et al. [47]	62.96

Table 4.1: Comparison of different reported results (per-video) on KTH dataset. There are many variations in terms of experiment setups among different methods, so a precise comparison between these methods is not possible.

sample tracked frames are shown in Fig. 4.8. Fig. 4.10(b) shows how our per-frame and per-video classification accuracies change as we vary the codebook size. S-LDA achieves a result of 100% per-video classification and 98.91% per-frame classification with 650 codewords. S-CTM achieves 100% accuracies for both per-frame and per-video classifications. In Fig. 4.10(a), we show the confusion matrix for per-frame classification of S-LDA with 650 codewords. We do not show the confusion matrix for per-video classification, since it is simply a perfect diagonal matrix. We compare our results with previous methods in Table 4.2. Again, we accept the fact that the comparison to Niebles & Fei-Fei [69] is not completely fair, since their method does not require any tracking or background subtraction. Also notice that [9] classifies space-time cubes. It is not clear how it can be compared with other methods that classify frames or videos.

Again, we conduct another experiment using an SVM classifier with the large-scale features for video classification. The per-video accuracy is 98.8% (linear SVM with $C = 1000$), which is inferior to our approaches (S-LDA/S-CTM).

Hockey dataset: The hockey dataset contains 70 tracks of hockey players performing eight actions (skate down, skate left, skate leftdown, skate leftup, skate right, skate rightdown, skate rightup, skate up). The action labels we obtained from the authors of [61] are

	per-frame(%)	per-video(%)	per-cube(%)
S-LDA	98.91	100	N/A
S-CTM	100	100	N/A
Fathi & Mori [26]	99.9	100	N/A
Jhuang et al. [45]	N/A	98.8	N/A
Niebles & Fei-Fei [69]	55	72.8	N/A
Blank et al. [9]	N/A	N/A	99.64

Table 4.2: Comparison of classification accuracies (per-frame, per-video, and per-cube) with previous methods on Weizmann dataset.

	per-frame(%)	per-video (%)
S-LDA	85.43	87.50
S-CTM	77.43	76.04

Table 4.3: Classification accuracies (per-frame and per-video) of S-LDA and S-CTM on the hockey dataset.

slightly different from what they used in [61]. Some sample frames are shown in Fig. 4.8. The confusion matrix of per-video classification of S-LDA is shown in Fig. 4.11(a). Again, most of the mistakes are intuitively reasonable, e.g., "skate leftdown" is confused with "skate down", "skate rightdown" is confused with "skate down", "skate rightup" is confused with "skate up", etc. In Fig. 4.11(b), we show how per-frame and per-video classification accuracies vary with different codebook sizes.

We compare the result of S-LDA and S-CTM in Table 4.3. The KNN method based on HOG features in Lu et al. [61] achieves a per-frame classification accuracy of 76.37%. However, since Lu et al. [61] use different class labels (they only have four class labels instead of eight) and experiment setup ($\sim$4000 frames for training, $\sim$1000 frames for testing, frames from a single video could be in both training and testing sets), a direct comparison with their approach is impossible.

Soccer dataset: The soccer dataset contains several minutes of digitized World Cup football game from an NTSC video tape. A preprocessing step is taken to track and stabilize each human figure. In the end, we obtain 35 video sequences, each corresponding to a person moving in the center of the field of view. We flip the sequences and get 70 video sequences in total. All the frames in these video sequences are hand-labeled with one of 8 action labels: “run left 45 °”, “run left”, “walk left”, “walk in/out”, “run in/out”, “walk right”, “run right”, “run right 45 °”. Representative frames of a single tracked person are shown in Fig. 4.8. The confusion matrix of S-LDA using 950 codewords is shown in Fig. 4.12(a). The overall accuracy is 77.81%. The overall accuracy of S-CTM is 78.64%. A previously reported result is in [24], which uses a k-nearest neighbor classifier based on the temporally smoothed motion feature vectors. However, the comparison with [24] is a bit tricky, since [24] only flips some sequences, and there is no way to figure out which sequences are flipped in their experiment. In order to do a fair comparison, we re-run the KNN algorithm in [24] using their own implementation on our dataset. Table 4.4 shows the overall accuracy and the main diagonals of the confusion matrices of our methods, compared with KNN[24]. We can see that our methods perform better by a large margin. A lot of the mistakes made by our algorithm make intuitive sense. For example, “run left 45 °” is confused with “run left” and “run in/out”, “walk right” is confused with “walk in/out” and “run right”, etc. Fig. 4.12(b) shows the effect of codebook size on the accuracy. The best accuracy peaks at around 950 codewords for both S-LDA and S-CTM.

We can visualize the learned parameter θ. Since there is a θ parameter for each video, we cannot show all of them. In Fig. 4.13, we show the θ parameter of two different videos in the dataset learned by S-LDA. It is obtained as the empirical mean of samples drawn from Dir(γ) (see Sec. 4.1.4 for more details). It is obvious that the learned θ correctly captures the topic proportions (i.e., the proportion of actions) in each video.

Ballet dataset: Finally, we test our algorithm on a ballet dataset we collected from an instructional ballet DVD. This dataset has been used in Fathi & Mori [26][1]. We obtain 44 tracks using a simple tracking algorithm based on color histograms. We manually label each frame with one of the eight action labels:“left-to-right hand opening”, “right-to-left hand opening”, “standing hand opening”, “leg swinging”, “jumping”, “turning”, “hopping”, and “standing still”. Some sample frames are shown in Fig. 4.8. Fig. 4.14(a) shows the confusion matrix of per-frame classification of S-LDA. Our algorithm classifies most actions correctly. The only exception is “standing still”. The reason is because it is difficult to reliably obtain optical flow features for this action. Fig. 4.14(b) shows how the classification accuracy varies with different codebook sizes.

We compare our results with Fathi & Mori [26] in Table 4.5. Since Fathi & Mori [26]

[1]The dataset is available at `http://www.cs.sfu.ca/research/groups/VML/semilatent/`

	S-LDA	S-CTM	KNN
run left 45°	0.4909	0.6208	0.3277
run left	0.8273	0.9557	0.6195
walk left	0.9149	0.8821	0.7585
walk in/out	0.8552	0.8552	0.4202
run in/out	0.7712	0.7390	0.2910
walk right	0.8821	0.8821	0.7585
run right	0.8076	0.8076	0.6195
run right 45°	0.6208	0.6208	0.3277
Overall	0.7781	0.7864	0.4923

Table 4.4: Comparison of the overall accuracy (per-frame) and the main diagonal of the confusion matrix in our method and the k-nearest neighbor (KNN) method in Efros et al. [24] on the soccer dataset. The result of KNN is obtained by running their own implementation on our dataset, see the text for details.

	per-frame(%)
S-LDA	88.66
S-CTM	91.36
Fathi & Mori [26]	51

Table 4.5: Comparison of the overall accuracy (per-frame) with Fathi & Mori [26] on the ballet dataset.

use exactly the same experiment setup, the comparison is fair. We can see that our method performs significantly better.

Remarks: From the above experiments, we can see that both S-LDA and S-CTM perform quite well. On the KTH and Weizmann datasets, the results of both models are quite similar. On the soccer and ballet datasets, S-CTM seems to perform better than S-LDA. This is reasonable, since these two datasets have multiple actions in a video sequence, S-LDA does not capture their correlations as well as S-CTM. On the hockey dataset, S-LDA seems to perform much better than S-CTM. We believe it is due to the fact that the size of the dataset is quite small, and there are no strong correlations among different class labels (the video only contains single actions). In this case, S-CTM is prone to overfitting since there are more model parameters to estimate.

We would like to point out that for single-action videos, one can train a discrimative classifier (e.g., SVM) to classify the videos. But our experimental results on KTH and Weizmann datasets show that S-LDA and S-CTM outperform SVM classifiers using the same large-scale features. The real advantage of our models is that we can deal with multiple actions in a video. Of course, one can also train a discriminative classifier to classify each individual frame, e.g., [24, 26], but our experimental results on the soccer and ballet datasets demonstrate empirically that those approaches perform inferior to ours.

Our experimental results also demonstrate that our approach is not very sensitive to the codebook size. The accuracy is quite stable in a large range of codebook sizes.

4.3 Summary

We have presented two supervised hierarchical topic models for action recognition based on motion words. Compared with previous topic models used in visual recognition, our models are different in several aspects. First, a "visual word" in our models is obtained from large-scale motion descriptors from a whole frame, rather than small space-time patches. The main motivation for using large-scale descriptors is that they better capture the characteristics of human actions. Second, the "latent topics" in our models directly correspond to different action categories. And class labels in the training data are naturally exploited in the learning process. On five different datasets, our methods consistently achieve superior results – either comparable to, or significantly better than other state-of-the-art results.

Of course, our method has its own limitations. For example, it requires a preprocessing stage of tracking and stabilizing human figures. However, we believe this is a reasonable assumption in many scenarios. In fact, all the video sequences in our experiments are pre-processed by off-the-shelf tracking/detection algorithms. The current datasets we use do not have significant background clutter. It will be interesting to explore the use of our method on more complicated datasets. As future work, we would like to collect and try our approach on more difficult datasets.

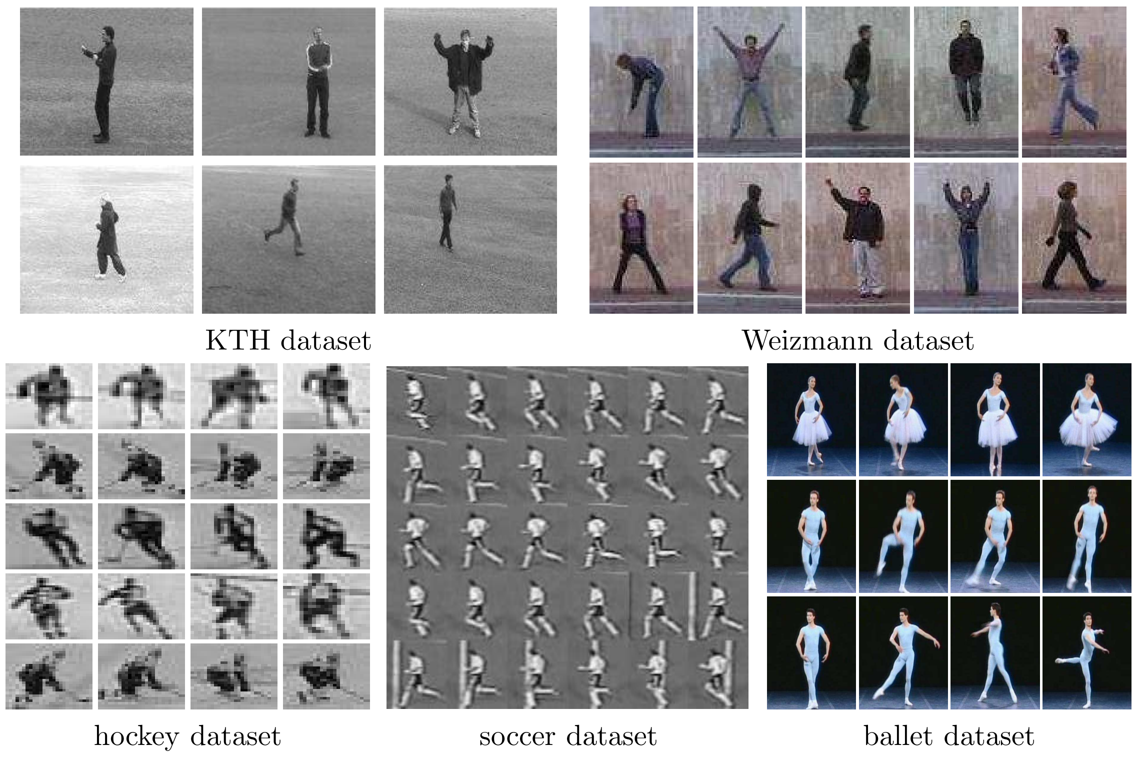
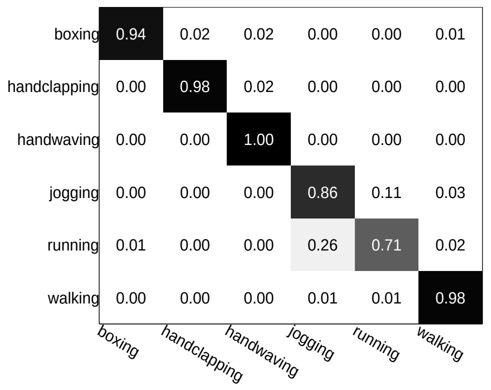
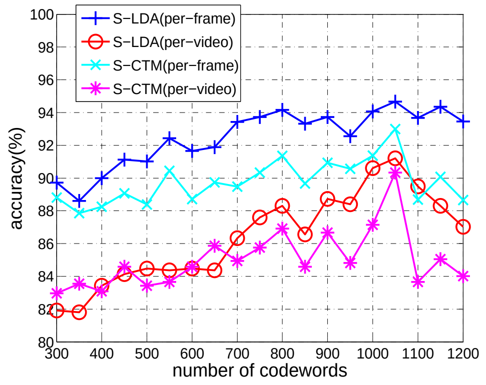


Figure 4.8: Sample frames from our datasets. The action labels in each dataset are as follows. (1) KTH dataset: walking, jogging, running, boxing, hand waving, hand clapping; (2) Weizmann dataset: running, walking, jumping-jack, jumping-forward-on-two-legs, jumping-in-place-on-two-legs, galloping-sideways, waving-two-hands, waving-one-hand, bending; (3) hockey dataset: skating down, skating left, skating leftdown, skating leftup, skating right, skating rightdown, skating rightup, skating up; (4) soccer dataset: run left 45°, run left, walk left, walk in/out, run in/out, walk right, run right, run right 45°; (5) ballet dataset: left-to-right hand opening, right-to-left hand opening, standing hand opening, leg swinging, jumping, turning, hopping, standing still.



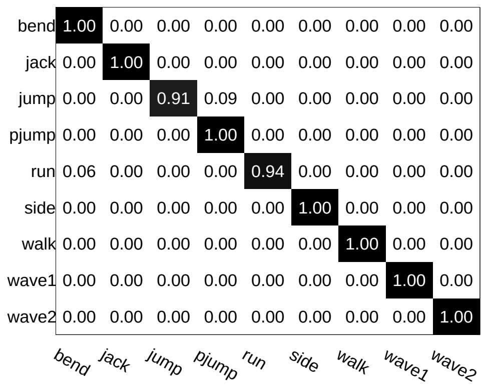
	boxing	handclapping	handwaving	jogging	running	walking
boxing	0.94	0.02	0.02	0.00	0.00	0.01
handclapping	0.00	0.98	0.02	0.00	0.00	0.00
handwaving	0.00	0.00	1.00	0.00	0.00	0.00
jogging	0.00	0.00	0.00	0.86	0.11	0.03
running	0.01	0.00	0.00	0.26	0.71	0.02
walking	0.00	0.00	0.00	0.01	0.01	0.98

(a)



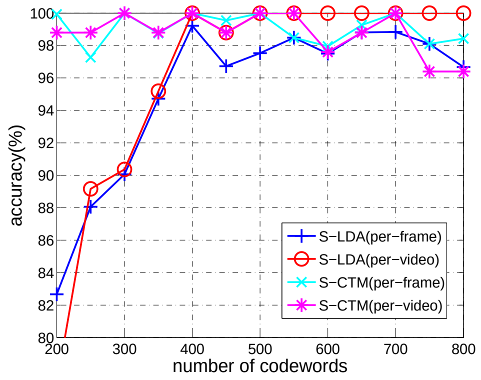
(b)

Figure 4.9: Results on KTH dataset: (a) confusion matrix for per-video classification of S-LDA using 1050 codewords (overall accuracy=91.2%). Horizontal rows are ground truths, and vertical columns are predictions; (b) classification accuracy (per-frame and per-video) vs. codebook size.



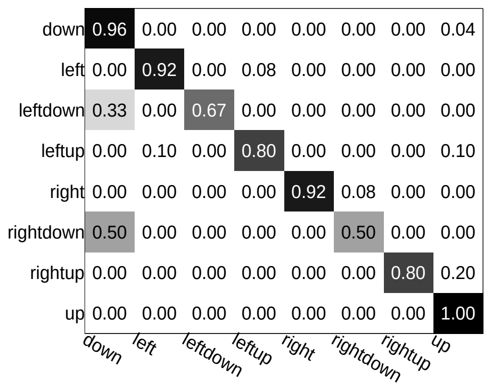
	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jack	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jump	0.00	0.00	0.91	0.09	0.00	0.00	0.00	0.00	0.00
pjump	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
run	0.06	0.00	0.00	0.00	0.94	0.00	0.00	0.00	0.00
side	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
walk	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
wave1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
wave2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

(a)



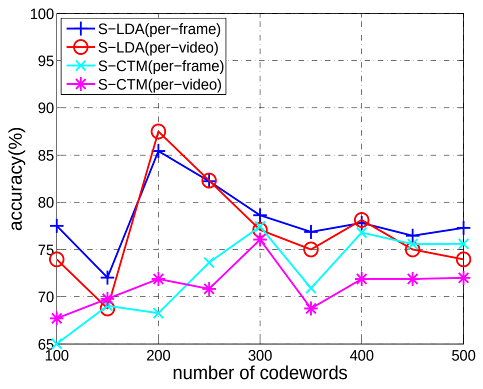
(b)

Figure 4.10: Results on Weizmann dataset: (a) confusion matrix for per-frame classification of S-LDA using 700 codewords (overall accuracy=98.83%); (b) classification accuracy (per-frame and per-video) vs. codebook size.



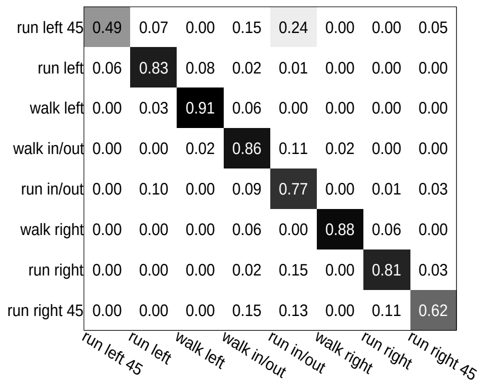
	down	left	leftdown	leftup	right	rightdown	rightup	up
down	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.04
left	0.00	0.92	0.00	0.08	0.00	0.00	0.00	0.00
leftdown	0.33	0.00	0.67	0.00	0.00	0.00	0.00	0.00
leftup	0.00	0.10	0.00	0.80	0.00	0.00	0.00	0.10
right	0.00	0.00	0.00	0.00	0.92	0.08	0.00	0.00
rightdown	0.50	0.00	0.00	0.00	0.00	0.50	0.00	0.00
rightup	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.20
up	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

(a)



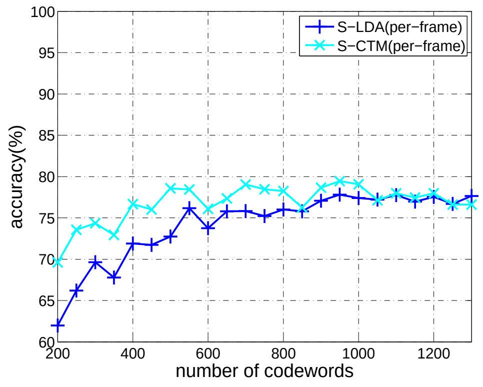
(b)

Figure 4.11: Results on hockey dataset: (a) confusion matrix for per-video classification of S-LDA using 200 codewords (overall accuracy=87.5%); (b) classification accuracy (per-frame and per-video) vs. codebook size.



	run left 45	run left	walk left	walk in/out	run in/out	walk right	run right	run right 45
run left 45	0.49	0.07	0.00	0.15	0.24	0.00	0.00	0.05
run left	0.06	0.83	0.08	0.02	0.01	0.00	0.00	0.00
walk left	0.00	0.03	0.91	0.06	0.00	0.00	0.00	0.00
walk in/out	0.00	0.00	0.02	0.86	0.11	0.02	0.00	0.00
run in/out	0.00	0.10	0.00	0.09	0.77	0.00	0.01	0.03
walk right	0.00	0.00	0.00	0.06	0.00	0.88	0.06	0.00
run right	0.00	0.00	0.00	0.02	0.15	0.00	0.81	0.03
run right 45	0.00	0.00	0.00	0.15	0.13	0.00	0.11	0.62

(a)



(b)

Figure 4.12: Results on soccer dataset: (a) confusion matrix for per-frame classification of S-LDA using 950 codewords (overall accuracy=77.81%); (b) per-frame classification accuracy vs. codebook size.

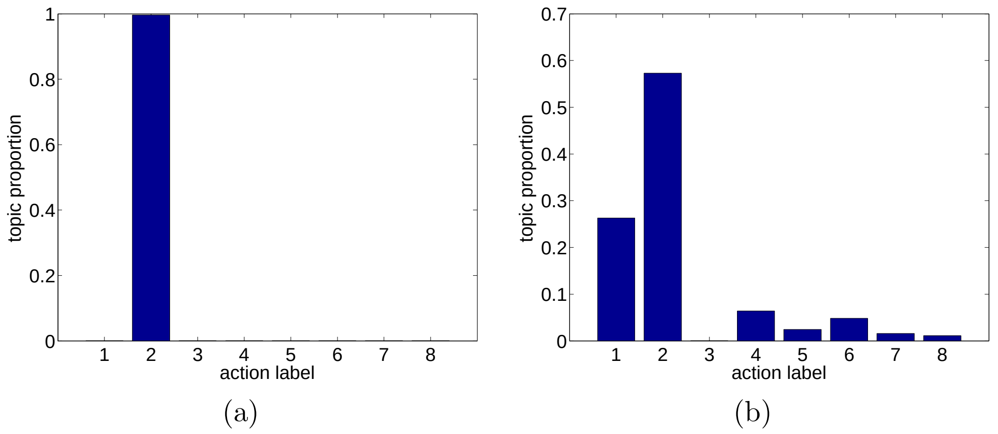
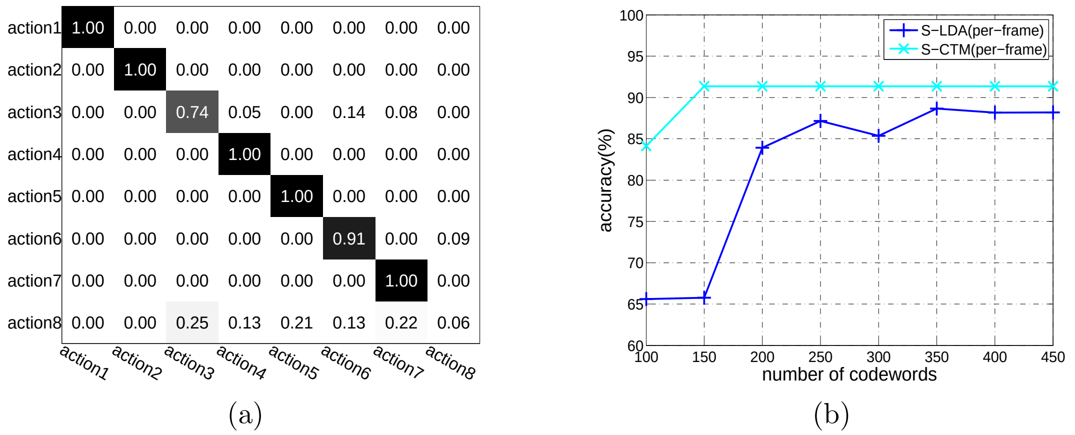


Figure 4.13: Visualization of θ parameters for two different videos in the soccer dataset: (a) a video with all the frames labeled as action 2(“run left”); (b) a video with all the frames labeled as action 1 (“run left 45°”) or 2 (“run left”).



	action1	action2	action3	action4	action5	action6	action7	action8
action1	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
action2	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
action3	0.00	0.00	0.74	0.05	0.00	0.14	0.08	0.00
action4	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
action5	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
action6	0.00	0.00	0.00	0.00	0.00	0.91	0.00	0.09
action7	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
action8	0.00	0.00	0.25	0.13	0.21	0.13	0.22	0.06

Figure 4.14: Results on ballet dataset: (a) confusion matrix for per-frame classification of S-LDA using 350 codewords (overall accuracy=88.66%), see the text for descriptions of these actions; (b) per-frame classification accuracy vs. codebook size.

Chapter 5

HCRF for Action Recognition

In this chapter, we develop a discriminatively trained hidden part model to represent human actions. Our model is inspired by the hidden conditional random field (hCRF) model [76] in object recognition.

In object recognition, there are three major representations: global template (rigid, e.g. [22], or deformable, e.g. [6]), bag-of-words [28, 35, 56, 95], and part-based [30, 29]. All three representations have been shown to be effective on certain object recognition tasks. In particular, recent work [29] has shown that part-based models outperform global templates and bag-of-words on challenging object recognition tasks.

A lot of the ideas used in object recognition can also be found in action recognition. For example, there is work [9] that treats actions as space-time shapes and reduce the problem of action recognition to 3D object recognition. In action recognition, both global template [24] and bag-of-words models [70, 23, 71] have been shown to be effective on certain tasks. Although conceptually appealing and promising, the merit of part-based models has not yet been widely recognized in action recognition. The goal of this work is to address this gap.

Our work is partly inspired by a recent work in part-based event detection [48]. In that work, template matching is combined with a pictorial structure model to detect and localize actions in crowded videos. One limitation of that work is that one has to manually specify the parts. Unlike Ke et al. [48], the parts in our model are initialized automatically.

The major contribution of this work is that we combine the flexibility of part-based approaches with the global perspectives of large-scale template features in a discriminative model. We show that the combination of part-based and large-scale template features improve the final results.

The rest of the chapter is organized as follows. Section 5.1 gives details of our model. Section 5.2 describes how we do learning and inference in the model. We present experimental results in Section 5.3 and conclude in Section 5.4.

5.1 Model

Our model is based on the hidden conditional random field (hCRF) model [76] proposed for object recognition. Objects are modeled as flexible constellations of parts conditioned on appearances of local patches found by interest point operators. The probability of the assignment of parts to local features is modeled by a conditional random field (CRF) [50]. The advantage of the hCRF is that it relaxes the conditional independence assumption commonly used in the bag-of-words approaches in object recognition.

Similarly, local patches can also be used to distinguish actions. Figure. 5.1 shows some examples of human motion and the local patches that can be used to distinguish them. A bag-of-words representation can be used to model these local patches for action recognition. However, it suffers from the same restriction of conditional independence assumption that ignores that the spatial structures of the parts. In this work, we use hCRF to model the constellation of these local patches.

There are also some important differences between objects and actions. For objects, local patches could carry enough information for recognition. But for actions, we believe local patches are not sufficiently informative. In our approach, we modify the hCRF model to combine local patches and large-scale global features.

We use the same optical flow features proposed by Efros et al. [24]. See Section 4.1.1 for details.

We use the hidden conditional random field (hCRF) [76] to model the images. Let I be a frame in a video sequence, $\mathbf{x}$ be the motion feature of this frame, and y be the corresponding class label of this frame, ranging over a finite label alphabet $\mathcal{Y}$. Our task is to learn a mapping from $\mathbf{x}$ to y. We assume each image I contains a set of salient patches $\{I_1, I_2, ..., I_m\}$. we will describe how to find these salient patches in Sec. 5.2.2. Our training set consists of labeled images $(\mathbf{x}^t, y^t)$ (as a notation convention, we use superscripts to index training images and subscripts to index patches) for $t = 1, 2, ..., n$, where $y^t \in \mathcal{Y}$ and $\mathbf{x}^t = (x_1^t, x_2^t ..., x_m^t)$. $x_i^t = \mathbf{x}^t(I_j^t)$ is the feature vector extracted from the global motion

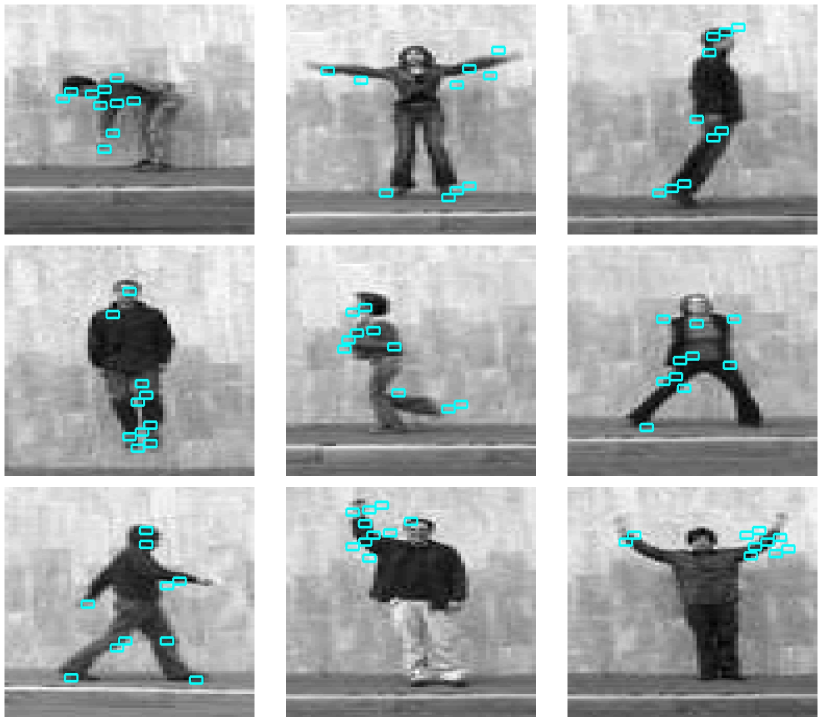


Figure 5.1: Examples of local discriminative patches. Each image shows a different human action. Each square indicates a local patch that is discriminative. See Sec. 5.2.2 for details about how these patches are obtained. In our model, we combine these local patch features with large-scale global features that capture the actions of the whole person.

feature $\mathbf{x}^t$ at the location of the patch I_j^t. For each image $I = \{I_1, I_2, ..., I_m\}$, we assume a vector of "part" variables $\mathbf{h} = \{h_1, h_2, ..., h_m\}$, where each h_i takes value from a finite set of possible set $\mathcal{H}$ of parts. Intuitively, each h_i assigns a part label to the patch I_i, where $i = 1, 2, ..., m$. For example, for the action "waving-two-hands", these parts may be used to characterize the movement patterns of the left and right arms. The values of $\mathbf{h}$ are not observed in the training set, and will become the hidden variables of the model.

We assume there are certain constraints between some pairs of (h_j, h_k). For example, in the case of "waving-two-hands", two patches h_j and h_k at the left hand might have the constraint that they tend to have the same part label, since both of them are characterized by the movement of the left hand. If we consider $h_i (i = 1, 2, ..., m)$ to be vertices in a graph $G = (E, V)$, the constraint between h_j and h_k is denoted by an edge $(j, k) \in E$. See Fig. 5.2 for an illustration of our model. Note that the graph structure can be different for different images. We will describe how to find the tree structure E in Section 5.2.2.

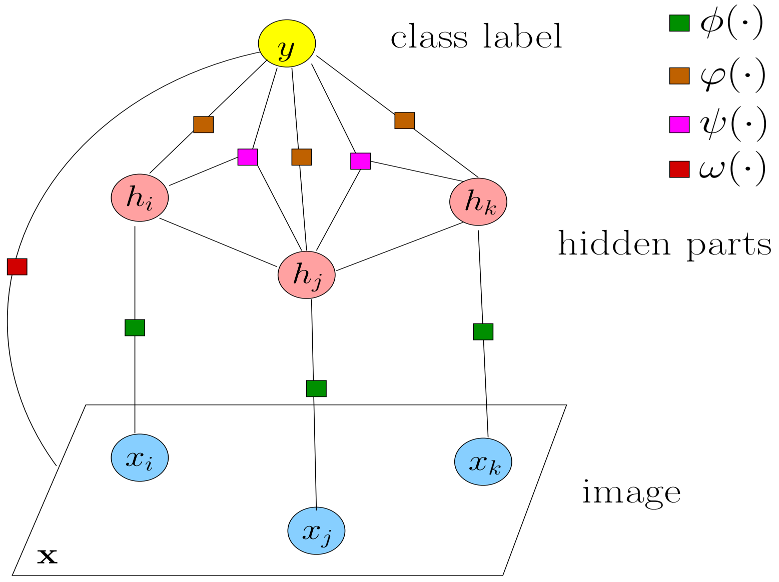


Figure 5.2: Illustration of the model. Each circle corresponds to a variable, and each square corresponds to a factor in the model.

Given the motion feature $\mathbf{x}$ of an image I, its corresponding class label y, and part labels $\mathbf{h}$, a hidden conditional random field is defined as follows:

$$p(y, \mathbf{h}|\mathbf{x}; \theta) = \frac{\exp(\Psi(y, \mathbf{x}, \mathbf{h}; \theta))}{\sum_{\hat{y}\in\mathcal{Y}} \sum_{\hat{\mathbf{h}}\in\mathcal{H}^m} \exp(\Psi(\hat{y}, \mathbf{x}, \hat{\mathbf{h}}; \theta))} \tag{5.1}$$

where θ is the model parameter, and $\Psi(y, \mathbf{h}, \mathbf{x}; \theta) \in \mathbb{R}$ is a potential function parameterized by θ. It follows that

$$p(y|\mathbf{x}; \theta) = \sum_{\mathbf{h}\in\mathcal{H}^\mathbf{m}} p(y, \mathbf{h}|\mathbf{x}; \theta) = \frac{\sum_{\mathbf{h}\in\mathcal{H}^m} \exp(\Psi(y, \mathbf{h}, \mathbf{x}; \theta))}{\sum_{\hat{y}\in\mathcal{Y}} \sum_{\mathbf{h}\in\mathcal{H}^m} \exp(\Psi(\hat{y}, \mathbf{h}, \mathbf{x}; \theta))} \tag{5.2}$$

We assume $\Psi(y, \mathbf{h}, \mathbf{x})$ is linear in the parameters $\theta = \{\alpha, \beta, \gamma, \eta\}$:

$$\begin{aligned}\Psi(y, \mathbf{h}, \mathbf{x}; \theta) \quad &= \quad \sum_{j\in V} \alpha^T \cdot \phi(x_j, h_j) + \sum_{j\in V} \beta^T \cdot \varphi(y, h_j) \\ &\quad + \sum_{(j,k)\in E} \gamma^T \cdot \psi(y, h_j, h_k) + \eta^T \cdot \omega(y, \mathbf{x})\end{aligned} \tag{5.3}$$

where $\phi(\cdot)$ and $\varphi(\cdot)$ are feature vectors depending on unary h_j's, $\psi(\cdot)$ is a feature vector depending on pairs of (h_j, h_k), $\omega(\cdot)$ is a feature vector that does not depend on the values of hidden variables. The details of these feature vectors are described in the following.

Unary potential $\alpha^T \cdot \phi(x_j, h_j)$ **:**

This potential function models the compatibility between x_j and the part label h_j, i.e., how likely the patch x_j is labeled as part h_j. It is parameterized as

$$\alpha^T \cdot \phi(x_j, h_j) = \sum_{c\in\mathcal{H}} \alpha_c^T \cdot \mathbf{1}_{\{h_j=c\}} \cdot [f^a(x_j)\ f^s(x_j)] \tag{5.4}$$

where we use $[f^a(x_j)\ f^s(x_j)]$ to denote the concatenation of two vectors $f^a(x_j)$ and $f^s(x_j)$. $f^a(x_j)$ is a feature vector describing the appearance of the patch x_j. In our case, $f^a(x_j)$ is simply the concatenation of four channels of the motion features at patch x_j, i.e., $f^a(x_j) = [Fb_x^+(x_j)\ Fb_x^-(x_j)\ Fb_y^+(x_j)\ Fb_y^-(x_j)]$. $f^s(x_j)$ is a feature vector describing the spatial location of the patch x_j. We discretize the whole image locations into l bins, and $f^s(x_j)$ is a length l vector of all zeros with a single one for the bin occupied by x_j. The parameter α_c^T can be interpreted as the measurement of compatibility between feature vector $[f^a(x_j)\ f^s(x_j)]$ and the part label $h_j = c$. The parameter α is simply the concatenation of α_c for all $c \in \mathcal{H}$.

Unary potential $\beta^T \cdot \varphi(y, h_j)$:

This potential function models the compatibility between class label y and part label h_j, i.e., how likely an image with class label y contains a patch with part label h_j. It is parameterized as

$$\beta^T \cdot \varphi(y, h_j) = \sum_{a \in \mathcal{Y}} \sum_{b \in \mathcal{H}} \beta_{a,b} \cdot \mathbf{1}_{\{y=a\}} \cdot \mathbf{1}_{\{h_j=b\}} \tag{5.5}$$

where $\beta_{a,b}$ indicates the compatibility between $y = a$ and $h_j = b$.

Pairwise potential $\gamma^T \cdot \psi(y, h_j, h_k)$:

This pairwise potential function models the compatibility between class label y and a pair of part labels (h_j, h_k), i.e., how likely an image with class label y contained a pair of patches with part labels h_j and h_k, where $(j, k) \in E$ corresponds an edge in the graph. It is parameterized as

$$\gamma^T \cdot \psi(y, h_j, h_k) = \sum_{a \in \mathcal{Y}} \sum_{b \in \mathcal{H}} \sum_{c \in \mathcal{H}} \gamma_{a,b,c} \cdot \mathbf{1}_{\{y=a\}} \cdot \mathbf{1}_{\{h_j=b\}} \cdot \mathbf{1}_{\{h_k=c\}} \tag{5.6}$$

where $\gamma_{a,b,c}$ indicates the compatibility of $y = a$, $h_j = b$ and $h_k = c$ for the edge $(j, k) \in E$.

Root model $\eta^T \cdot \omega(y, \mathbf{x})$:

The root model is a potential function that models the compatibility of class label y and the large-scale global feature of the whole image. It is parameterized as

$$\eta^T \cdot \omega(y, \mathbf{x}) = \sum_{a \in \mathcal{Y}} \eta_a^T \cdot \mathbf{1}_{\{y=a\}} \cdot g(\mathbf{x}) \tag{5.7}$$

where $g(\mathbf{x})$ is a feature vector describing the appearance of the whole image. In our case, $g(\mathbf{x})$ is the concatenation of all the four channels of the motion features in the image, i.e., $g(\mathbf{x}) = [Fb_x^+ \ Fb_x^- \ Fb_y^+ \ Fb_y^-]$. η_a can be interpreted as a root filter that measures the compatibility between the appearance of an image $g(\mathbf{x})$ and a class label $y = a$. And η is simply the concatenation of η_a for all $a \in \mathcal{Y}$.

The parameterization of $\Psi(y, \mathbf{h}, \mathbf{x})$ is similar to that used in object recognition [76]. But there are two important differences. First of all, our definition of the unary potential function $\phi(\cdot)$ encodes both appearance and spatial information of the patches. Secondly, we have a potential function $\omega(\cdot)$ describing the large scale appearance of the whole image. The representation in Quattoni et al. [76] only models local patches extracted from the image. This may be appropriate for object recognition. But for human action recognition, it is not clear that local patches can be sufficiently informative. We will demonstrate this experimentally in Section 5.3.

5.2 Learning and Inference

The model parameters θ are learned by maximizing the conditional log-likehood on the training images:

$$\begin{aligned} \theta^* &= \arg\max_\theta L(\theta) = \arg\max_\theta \sum_t \log p(y^t|\mathbf{x}^t;\theta) && (5.8) \\ &= \arg\max_\theta \sum_t \log\left(\sum_{\mathbf{h}} p(y^t,\mathbf{h}|\mathbf{x}^t;\theta)\right) && (5.9) \end{aligned}$$

The objective function $L(\theta)$ in Quattoni et al.[76] also has a regularization term $\frac{-1}{2\sigma^2}||\theta||^2$. In our experiments, we find that the regularization does not seem to have much effect on the final results, so we will use the un-regularized version. Different from conditional random field (CRF) [50], the objective function $L(\theta)$ of hCRF is not concave, due to the hidden variables $\mathbf{h}$. But we can still use gradient ascent to find θ that is locally optimal. The gradient of the log-likelihood with respect to the t-th training image $(\mathbf{x}^t, y^t)$ can be calculated as:

$$\begin{aligned} \frac{\partial L^t(\theta)}{\partial\alpha} &= \sum_{j\in V}\left[\mathbb{E}_{p(h_j|y^t,\mathbf{x}^t;\theta)}\phi(x_j^t,h_j) - \mathbb{E}_{p(h_j,y|\mathbf{x}^t;\theta)}\phi(x_j^t,h_j)\right] \\ \frac{\partial L^t(\theta)}{\partial\beta} &= \sum_{j\in V}\left[\mathbb{E}_{p(h_j|y^t,\mathbf{x}^t;\theta)}\varphi(h_j,y^t) - \mathbb{E}_{p(h_j,y|\mathbf{x}^t;\theta)}\varphi(h_j,y)\right] \\ \frac{\partial L^t(\theta)}{\partial\gamma} &= \sum_{(j,k)\in E}\left[\mathbb{E}_{p(h_j,h_k|y^t,\mathbf{x}^t;\theta)}\psi(y^t,h_j,h_k) - \mathbb{E}_{p(h_j,h_k,y|\mathbf{x}^t;\theta)}\psi(y,h_j,h_k)\right] \\ \frac{\partial L^t(\theta)}{\partial\eta} &= \omega(y^t,\mathbf{x}^t) - \mathbb{E}_{p(y|\mathbf{x}^t;\theta)}\omega(y,\mathbf{x}^t) && (5.10) \end{aligned}$$

5.2.1 Computing feature expectations

In order to calculate the feature expectation needed for the gradient, we need to compute the following conditional marginal probabilities $p(h_j|\mathbf{x}^t, y^t)$, $p(h_j, y|\mathbf{x}^t)$, $p(h_j, h_k|\mathbf{x}^t, y^t)$, $p(h_j, h_k, y|\mathbf{x}^t)$, $p(y|\mathbf{x}^t)$. Assuming the edges E form a tree, these conditional marginals can be calculated exactly in $O(|\mathcal{Y}||E||\mathcal{H}|^2)$ time using belief propagation.

To compute $p(h_j|\mathbf{x}^t, y^t)$ and $p(h_j, h_k|\mathbf{x}^t, y^t)$, the set of "upstream" messages from part j to its parent k are computed as follows.

$$m_j(h_k) \propto \sum_{h_j} \exp\left[\gamma^T \cdot \psi(y^t, h_j, h_k)\right] \cdot a_j(h_j) \tag{5.11}$$

$$a_j(h_j) \propto \exp\left[\alpha^T \cdot \phi(x_j, h_j) + \beta^T \cdot \varphi(y^t, h_j)\right] \prod_{i \in \text{kids}_j} m_i(h_j) \tag{5.12}$$

At the root r, a_r is the true conditional marginal $p(h_r|\mathbf{x}^t, y^t)$ (up to a proper normalization). Starting from the root, we pass messages downstream from k to j to compute the remaining marginals. We also simultaneously compute expectations over pairwise marginals:

$$p(h_j|\mathbf{x}, y^t) \propto a_j(h_j) \sum_{h_k} \exp\left[\gamma^T \cdot \psi(y^t, h_j, h_k)\right] p(h_k|\mathbf{x}, y^t) \tag{5.13}$$

$$\mathbb{E}(\psi(y^t, h_j, h_k)) \propto \sum_{h_k} p(h_k|\mathbf{x}^t, y^t) \sum_{h_j} \psi(y^t, h_j, h_k) \exp\left[\gamma^T \cdot \psi(y^t, h_j, h_k)\right] a_j(h_j)$$

The marginals $p(h_j, y|\mathbf{x}^t)$ and $p(h_j, h_k, y|\mathbf{x}^t)$ can be calculated simultaneously in a similar way. We just need to modify the above message passing scheme in two ways. Firstly, we will maintain and pass the messages separately for each possible value of y. Secondly, the upstream message for the root node r is modified as following to take into account the root filter response $\eta^T \cdot \omega(y, \mathbf{x})$:

$$m_j(y, h_r) \propto \sum_{h_j} \exp\left[\gamma^T \cdot \psi(y, h_j, h_r)\right] a_j(y, h_j) \tag{5.14}$$

$$a_j(y, h_j) \propto \exp\left[\alpha^T \cdot \phi(x_j, h_j) + \beta^T \cdot \varphi(y, h_j) + \eta^T \cdot \omega(y, \mathbf{x})\right] \prod_{i \in \text{kids}_j} m_i(y, h_j)$$

All the feature expectations in Eq. 5.10 can be easily obtained once the marginals are obtained.

5.2.2 Implementation details

Now we describe several details about how the above ideas are implemented.

Learning root filter η: Given a set of training images $(\mathbf{x}^t, y^t)$, we firstly learn the root filter η by solving the following optimization problem:

$$\eta^* = \arg\max_{\eta} \sum_t \log \mathcal{L}(y^t|\mathbf{x}^t;\eta) = \arg\max_{\eta} \sum_t \log \frac{\exp\left(\eta^T \cdot \omega(y^t, \mathbf{x}^t)\right)}{\sum_y \exp\left(\eta^T \cdot \omega(y, \mathbf{x}^t)\right)} \tag{5.15}$$

In other words, η^* is learned by only considering the feature vector $\omega(\cdot)$. We then use η^* as the starting point for η in the gradient ascent (Eq. 5.10). Other parameters α, β, γ are initialized randomly.

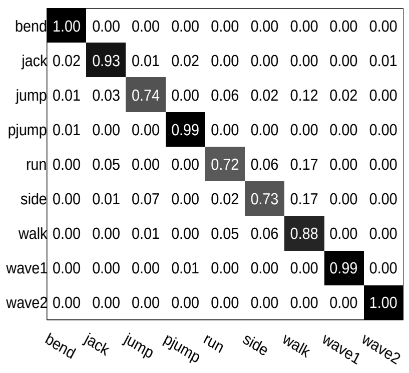
Patch initialization: We use a simple heuristic similar to that used in [29] to initialize ten salient patches on every training image from the root filter η^* trained above. For each training image I with class label a, we apply the root filter η_a on I, then select an rectangle region of size 5×5 in the image that has the most positive energy. We zero out the weights in this region and repeat until ten patches are selected. Figure 5.1 shows examples of the patches found in some images. The tree $G = (V, E)$ is formed by running a minimum spanning tree algorithm over the ten patches.

Inference: During testing, we do not know the class label of a given test image, so we cannot use the patch initialization described above to initialize the patches, since we do not know which root filter to use. Instead, we run root filters from all the classes on a test image, then calculate the probabilities of all possible instantiations of patches under our learned model, and classify the image by picking the class label that gives the maximum of the these probabilities. In other words, for a testing image with motion descriptor $\mathbf{x}$, we first obtain $|\mathcal{Y}|$ instances $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(|\mathcal{Y}|)}\}$, where each $\mathbf{x}^{(k)}$ is obtained by initializing the patches on $\mathbf{x}$ using the root filter η_k. The final class label y^* of $\mathbf{x}$ is obtained as $y^* = \arg\max_y \left[\max\{p(y|\mathbf{x}^{(1)};\theta), p(y|\mathbf{x}^{(2)};\theta), ..., p(y|\mathbf{x}^{(|\mathcal{Y}|)};\theta)\}\right]$.

5.3 Experiments

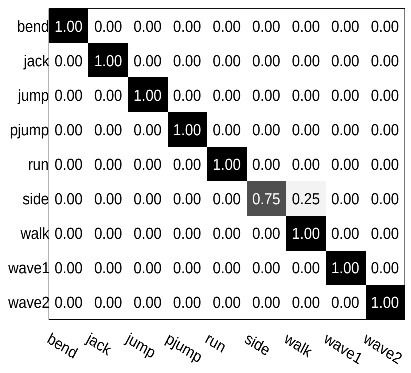
We test our algorithm on the KTH human motion dataset [90] and the Weizmann human action dataset [9].

Weizmann dataset: We randomly choose videos of five subjects as training set, and the videos in the remaining four subjects as test set. We learn three hCRF models with



	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jack	0.02	0.93	0.01	0.02	0.00	0.00	0.00	0.00	0.01
jump	0.01	0.03	0.74	0.00	0.06	0.02	0.12	0.02	0.00
pjump	0.01	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00
run	0.00	0.05	0.00	0.00	0.72	0.06	0.17	0.00	0.00
side	0.00	0.01	0.07	0.00	0.02	0.73	0.17	0.00	0.00
walk	0.00	0.00	0.01	0.00	0.05	0.06	0.88	0.00	0.00
wave1	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.99	0.00
wave2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Frame-by-frame classification



	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jack	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jump	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
pjump	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
run	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
side	0.00	0.00	0.00	0.00	0.00	0.75	0.25	0.00	0.00
walk	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
wave1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
wave2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Video classification

Figure 5.3: Confusion matrices of classification results on Weizmann dataset. Horizontal rows are ground truths, and vertical columns are predictions.

method	root model	local hCRF			our approach		
		$\|\mathcal{H}\| = 6$	$\|\mathcal{H}\| = 10$	$\|\mathcal{H}\| = 20$	$\|\mathcal{H}\| = 6$	$\|\mathcal{H}\| = 10$	$\|\mathcal{H}\| = 20$
per-frame	0.7470	0.5722	0.6656	0.6383	**0.8682**	**0.9029**	**0.8557**
per-video	0.8889	0.5556	0.6944	0.6111	**0.9167**	**0.9722**	**0.9444**

Table 5.1: Comparison of two baseline systems with our approach on Weizmann dataset.

different sizes of possible part labels, $|\mathcal{H}| = 6, 10, 20$. Our model classifies every frame in a video sequence (i.e., per-frame classification), but we can also obtain the class label for the whole video sequence by majority voting of the labels of its frames (i.e., per-video classification). We show the confusion matrix with $|\mathcal{H}| = 10$ for both per-frame and per-video classification in Fig. 5.3.

We compare our system to two baseline methods. The first baseline (root model) only uses the root filter $\eta^T \cdot \omega(y, \mathbf{x})$, which is simply a discriminative version of Efros et al. [24]. The second baseline (local hCRF) is similarly to our model, but without the root filter $\eta^T \cdot \omega(y, \mathbf{x})$, i.e., local hCRF only uses the root filter to initialize the salient patches, but does not use it in the final model. The comparative results are shown in Table 5.1. We can see that our approach is significantly better than the two baseline systems. We also compare our results (with $|\mathcal{H}| = 10$) with the results reported in [69] in Table 5.2. Our classification accuracy is significantly better, although we accept the fact that the comparison is not completely fair, since Niebles & Fei-Fei [69] does not require any tracking or background subtraction.

We visualize the learned parts in Fig. 5.4(a). Each patch is represented by a color that

	per-frame(%)	per-video(%)
Our method	**90.3**	**97.2**
Niebles & Fei-Fei [69]	55	72.8

Table 5.2: Comparison of classification accuracy (per-frame and per-video) with [69] on the Weizmann dataset.

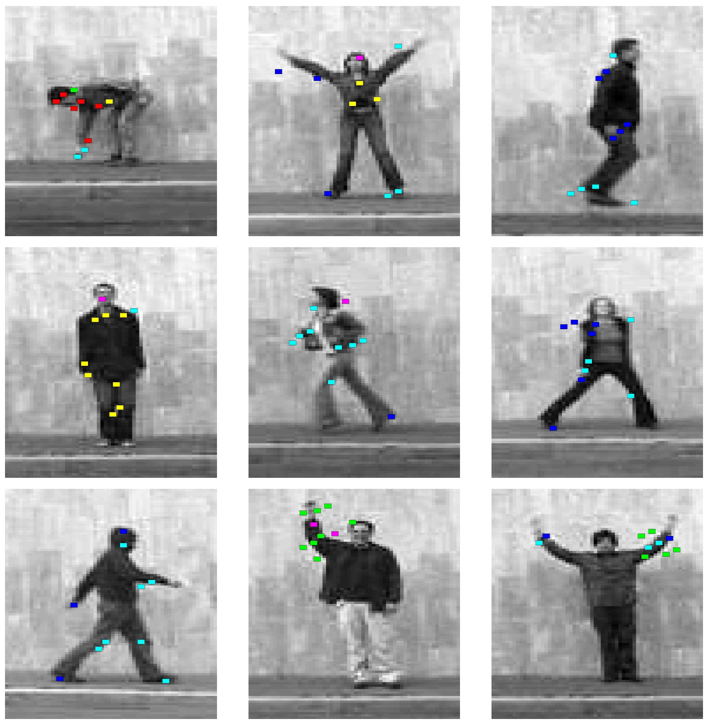
method	root model	local hCRF			our approach		
		$\|\mathcal{H}\| = 6$	$\|\mathcal{H}\| = 10$	$\|\mathcal{H}\| = 20$	$\|\mathcal{H}\| = 6$	$\|\mathcal{H}\| = 10$	$\|\mathcal{H}\| = 20$
per-frame	0.5377	0.4749	0.4452	0.4282	**0.6633**	**0.6698**	**0.6444**
per-video	0.7339	0.5607	0.5814	0.5504	**0.7855**	**0.8760**	**0.7512**

Table 5.3: Comparison of two baseline systems with our approach on KTH dataset.

corresponds to the most likely part label of that patch. We also visualize the root filters applied on these images in Fig. 5.4(b).

KTH dataset: Due to the large size of this dataset, we cannot perform our experiment on the whole dataset because of computation and memory constraints. Instead, we randomly choose ten frames from each video as our dataset. We randomly split the videos roughly equally into training and test sets. We make sure videos from the same subject doing the same action are not present in both training and test sets. The confusion matrices (with $|\mathcal{H}| = 10$) for both per-frame and per-video classification are shown in Fig. 5.5. The comparison with the two baseline algorithms is summarized in Table 5.3. Again, our approach outperforms the two baselines systems.

We have not found any previously published results on per-frame classification on KTH dataset. We can only compare with previous work in terms of per-video classification. The result is summarized in Table 5.4. We should emphasize that we do not attempt a direct comparison, since we did not use the whole dataset. In addition, different methods listed in Table 5.4 have all sort of variations in their experiments (e.g., different split of training/test data, whether temporal smoothing is used, whether per-frame classification can be performed, whether tracking/background subtraction is used, etc.), which make it impossible to directly compare them. We provide the results only to show that our approach is comparable to the state-of-the-art.


(a)

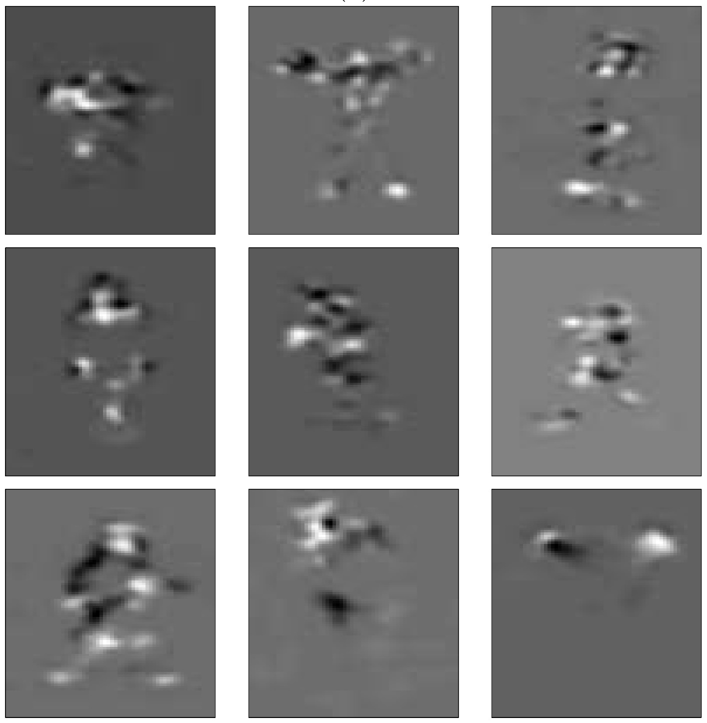
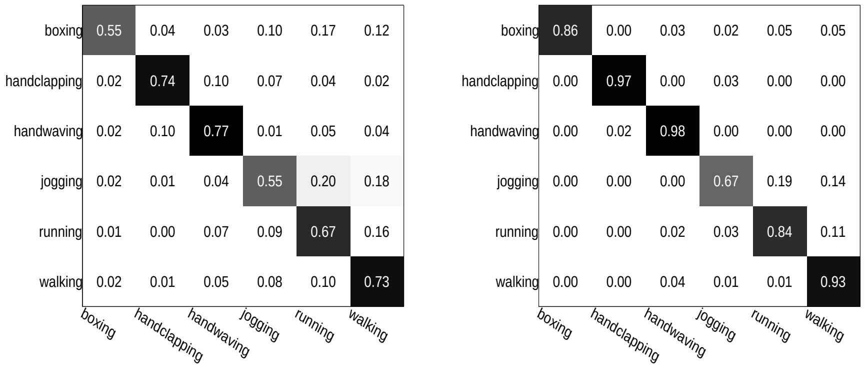

(b)

Figure 5.4: (a) Visualization of the learned parts. Patches are colored according to their most likely part labels. Each color corresponds to a part label. Some interesting observations can be made. For example, the part label represented by red seems to correspond to the "moving down" patterns mostly observed in the "bending" action. The part label represented by green seems to correspond to the motion patterns distinctive of "hand-waving" actions; (b) Visualization of root filters applied on these images. For each image with class label c, we apply the root filter η_c. The results show the filter responses aggregated over four motion descriptor channels. Bright areas correspond to positive energies, i.e., areas that are discriminative for this class.



Frame-by-frame classification　　Video classification

Figure 5.5: Confusion matrices of classification results on KTH dataset. Horizontal rows are ground truths, and vertical columns are predictions.

methods	%
Our method	**87.60**
Jhuang et al. [45]	**91.70**
Nowozin et al. [71]	87.04
Niebles et al. [70]	81.50
Dollár et al. [23]	81.17
Schuldt et al. [90]	71.72
Ke et al. [47]	62.96

Table 5.4: Comparison of per-video classification accuracy with previous approaches on KTH dataset.

5.4 Summary

We have presented a discriminatively learned part model for human action recognition. Unlike previous work [48], our model does not require manual specification of the parts. Instead, the parts are initialized by a learned root filter. Our model combines both large-scale features used in global templates and local patch features used in bag-of-words models. Our experimental results show that our model is quite effective in recognizing actions. The results are comparable to the state-of-the-art approaches.

Chapter 6

MMHCRF for Action Recognition

In this chapter, we develop a max-margin learning framework for solving various classification problems involving hidden structures. In particular, we apply our learning method to learn the HCRF model for action recognition in Chapter 5.

Many problems in computer vision can be formulated as *classification with structured latent variables*. Consider the following three vision tasks. (1) *Pedestrian detection*: it can be formulated as a binary classification problem that classifies an image patch x to be +1 (pedestrian) or -1(non-pedestrian). The locations of the body parts can be considered as latent variables in this case, since most of the existing training datasets for pedestrian detection do not provide this information. These latent variables are also *structured* – e.g., the location of the torso imposes certain constraints on the possible locations of other parts. Previous approaches [29, 106] usually use a tree structured model to model these constraints. (2) *Object recognition*: this problem is to assign a class label to an image that contains the object of interest. If we consider the figure/ground labeling of pixels of the image as latent variables, object recognition is also a problem of classification with latent variables. The latent variables are also structured, typically represented by a grid-structured graph. (3) *Object identification*: given two images, the task is to decide whether these are two images of the same object or not. If an image is represented by a set of patches found by interest point operators, one particular way to solve this problem is to first find the correspondence between patches in the two images, then learn a binary classifier based on the result of the correspondence [5]. Of course, the correspondence information is "latent" – not available in the training data, and "structured" – assuming one patch in one image matches to one or zero patches in the other image, this creates a combinatorial structure [103].

One simplified approach to solve the above-mentioned problems is to ignore the latent structures, and treat them as standard classification problems, e.g., Dalal & Triggs [22] in the case of pedestrian detection. However, there is evidence [29, 77, 106, 113] showing that incorporating latent structures into the system can improve the performance.

One way to solve problems of classification with structured latent variables is to use the HCRF model presented in Chapter 5. However, the difficulty of applying HCRF is that the training algorithm involves summing over all the latent variables. If the latent variables form certain structures (e.g. trees), this summation can be done exactly. But for a lot of interesting structures, this summation turns out to be a computational bottleneck. In this chapter, we propose a max-margin learning framework for training classifiers with structured latent variables that can be applied in a wider range of structures. We introduce an efficient learning algorithm based on the cutting plane method and decomposed dual optimization. We apply our proposed model to recognize human actions from video sequences, where we model a person by a root template and a constellation of several "parts", similar to the HCRF model in Chapter 5. We call our approach the *Max-Margin Hidden Conditional Random Field (MMHCRF).* We show experimentally that the MMHCRF achieves better performance than the HCRF. More importantly, the max-margin learning allows us to deal with more complex latent structures (e.g., matching/correspondence mentioned before).

In this chapter, we only consider latent structures that form a tree-structured model, and we restrict ourselves to the application domain of human action recognition. But the proposed framework is quite general and can be applied in other domains that involve highly complex latent structures.

MMHCRF is closely related to the hidden conditional random field (HCRF) model [77]. The basic idea of HCRF in object recognition is to model an object as a constellation of "parts" conditioned on local observations found by an interest point operator. For each object class, the probability of assigning part labels to local features is modeled by a conditional random field (CRF) [50]. The parameters of a HCRF model are learned by maximizing the conditional likelihood of the training data. The limitation of HCRFs is that the training involves summing over all the possible labelings of the latent variables. If the latent variables form certain graph structures (e.g. trees, graphs of low tree-width), this summation can be done analytically. For latent structures of general graph topology, HCRFs have to resort to various approximations (e.g. loopy belief propagation, mean-field variational methods, etc.) for training. For even more complicated structures (e.g. matching/correspondence), it is not even clear how to approximate the computation. Compared with HCRFs, our model can handle a much wider range of latent structures. This is in analog to the advantage of the max-margin Markov network (M^3N) [102, 103] over the CRF in the structured-output learning literature.

Another closely related work is the latent SVM (LSVM) [29]. The LSVM is proposed for object detection (binary classification). It models an object as a global coarse template covering an entire object and several higher resolution part templates. The positions of the root template and the parts are latent in LSVMs. LSVMs learn the model parameters using a max-margin discriminative training method. Our work is different from LSVMs in several directions. First of all, our model directly handles multi-class classification. Unfortunately, unlike LSVMs, the resultant optimization problem is not in a standard form that can be solved by off-the-shelf SVM solvers (e.g., SVMLight). We propose an efficient algorithm based on the cutting-plane method and the decomposed dual optimization. Our learning algorithm is intuitive and easy to implement. It is also quite general – one only needs to change a small component of the algorithm in order to handle different latent structures. The potential functions and parametrization of our model are also significantly different from those used in LSVMs. In our model, a hidden variable corresponds to a "part label", while in LSVMs, a hidden variable corresponds to the location of a part. The parts in our model can have a variety of constraints, while in the model used in LSVMs, the parts only have constraints with respect to a root filter.

6.1 Model

Our approach uses the same motion features used in Chapter 4 and Chapter 5. Please refer to Section 4.1.1 for details. We model a frame in a video in a similar way to the HCRF model presented in Chapter 5. For completeness, we briefly review the model in this section. A frame is represented by a global motion feature extracted from the whole frame and a set of salient local patches. Our model consists of a root filter and a constellation of several hidden parts. The root filter models the compatibility of the action label and the global motion feature of the whole frame. A hidden part assigns a latent "part label" to a local patch. Intuitively, those "part labels" correspond to local motion patterns that are useful for discriminating different actions. Figure 6.1 shows some sample frames and the learned hidden part labels. The constraints among the patches are captured by pairwise potential functions defined in the model.

Let I be a frame in a video sequence. We assume I contains a set of salient patches $\{I_1, I_2, ..., I_m\}$. Let x be the feature vector extracted from the frame I, and $y \in \mathcal{Y}$ be a class label, with discrete domain $\mathcal{Y}$. We assume x has the form of $x = (s_0, s_1, ..., s_m)$, where s_0 is the motion feature vector extracted from the whole frame, and $s_i (i = 1, 2, ..., m)$ is the motion feature vector extracted from the patch I_i. We assume I is associated with a vector of hidden variables of the form $h = (z_1, z_2, ..., z_m)$, where z_i takes values from a discrete set $\mathcal{H}$ of possible "part" labels. Intuitively, z_i assigns a "part label" to the patch I_i, where a "part label" corresponds to certain motion patterns distinctive of certain actions. For example, one "part label" might corresponds to the "moving down" pattern commonly observed in "bending" action, another "part label" might correspond to the motion pattern distinctive of "hand-waving" action. See Figure 6.1 for examples of learned part labels. In our model, there are certain constraints between some pairs of (z_j, z_k). We use a undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to represent $(z_1, z_2, ..., z_m)$, where a vertex $v_i \in \mathcal{V}$ corresponds to the part label z_i, and an edge $(v_j, v_k) \in \mathcal{E}$ corresponds to the constraint between z_j and z_k. Similar to HCRFs [77, 113], $\mathcal{E}$ is assumed to form a tree. The tree structure is obtained by running a minimum spanning tree algorithm over the salient patches.

The $\langle x, y \rangle$ pair is scored by a function of the form $f_w(x, y) = \max_h w^\top \Phi(x, h, y)$. Here w is a vector of model parameters and h are the latent variables. The example x is classified according to $y^* = \arg\max_y f_w(x, y)$. The model parameters w have four parts $w = \{w_0, w_1, w_2, w_3\}$, and $w^\top \Phi(x, h, y)$ is defined as:

$$\begin{aligned} w^\top \Phi(x, h, y) &= w_0^\top \cdot \phi_0(y, s_0) + \sum_{j \in \mathcal{V}} w_1^\top \cdot \phi_1(s_j, z_j) \\ &+ \sum_{j \in \mathcal{V}} w_2^\top \cdot \phi_2(y, z_j) + \sum_{(j,k) \in \mathcal{E}} w_3^\top \cdot \phi_3(y, z_j, z_k) \end{aligned} \tag{6.1}$$

The forms of these potential functions and the parametrization are identical to those in Chapter 5. $w_0^\top \phi_0(y, s_0)$ is a root filter that models the compatibility of the class label y and the large-scale global feature of the whole image. It is parametrized as $w_0^\top \phi_0(y, s_0) = \sum_{a \in \mathcal{Y}} w_{0a}^\top \cdot \mathbb{1}(y = a) \cdot g(s_0)$, where $g(s_0)$ is the concatenation of the motion features Fb_x^+, Fb_x^-, Fb_y^+, Fb_y^- from the patch s_0 (i.e., the whole image), w_{0a} can be interpreted as a root filter that measures the compatibility of the class label a and the motion vector $g(s_0)$, w_0 is a simply the concatenation of w_{0a} for $\forall a \in \mathcal{Y}$. Similarly, $w_1^\top \phi_1(s_j, z_j) = \sum_{c \in \mathcal{H}} w_{1c}^\top \cdot \mathbb{1}(z_j = c) \cdot g(s_j)$ measures the compatibility of the feature vector $g(s_j)$ extracted from the j-th

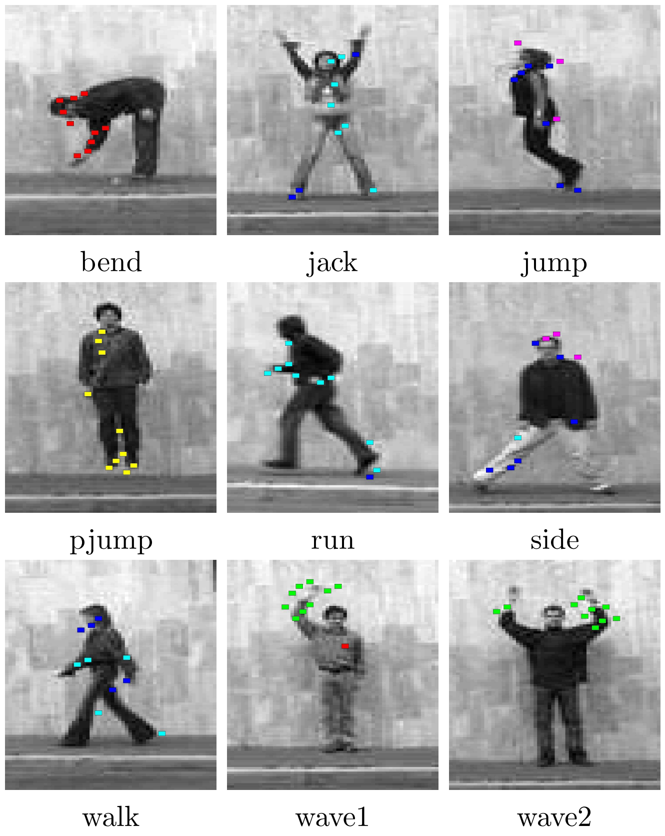


Figure 6.1: Visualization of the learned parts. Patches are colored according to their most likely part labels. Interesting observations can be made. For example, the parts colored in red, yellow and green seem to be distinctive of “bend” “pjump” and “wave” actions, respectively. We can also observe the “sharing of parts” among different actions. For example, the parts colored in blue and light blue are shared by “walk” “run” “jack” actions.

patch and a hidden part label z_j for the j-th patch. $w_2^\top \phi_2(y, z_j) = \sum_{a\in\mathcal{Y}} \sum_{b\in\mathcal{H}} w_{2ab} \cdot \mathbb{1}(y = a) \cdot \mathbb{1}(z_j = b)$ measures the compatibility of the class label y and a hidden part label z_j for the j-the patch. $w_3^\top \phi_3(y, z_j, z_k) = \sum_{a\in\mathcal{Y}} \sum_{b\in\mathcal{H}} \sum_{c\in\mathcal{H}} w_{3abc} \cdot \mathbb{1}(y = a) \cdot \mathbb{1}(z_j = b) \cdot \mathbb{1}(z_k = c)$ measures the compatibility of a class label y and a pair of hidden part labels (z_j, z_k) for a pair of (j, k)-th patches. These constraints defined on edges capture the intuition that patches with certain hidden part labels tend to appear together for certain actions.

6.2 Learning

Our learning method is inspired by the success of max-margin methods in machine learning [3, 19, 29, 102]. Given a learned model, the classification is achieved by first finding the best labeling of the hidden parts for each action, then picking the action label with the highest score. The learning algorithm aims to set the model parameters so that the scores of correct action labels on the training data are higher than the scores of incorrect action labels by a large margin.

6.2.1 Max-Margin HCRF

A max-margin hidden conditional random field (MMHCRF) is defined as follows. Recall that an $\langle x, y \rangle$ pair is scored by a function of the form:

$$f_w(x, y) = \max_h w^\top \Phi(x, h, y) \tag{6.2}$$

where w is the model parameter and h is a vector of hidden variables. In our work, we consider the case in which $h = (z_1, z_2, ..., z_m)$ forms a tree-structured undirected graphical model, but our proposed model is a rather general framework and can be applied to a wide variety of structures. We will briefly discuss them in Sec. 6.2.4. Similar to latent SVMs, MMHCRFs are instances of the general class of energy-based models [57].

Let $D = (\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, ..., \langle x_N, y_N \rangle)$ be a set of labeled training examples. The goal of learning is to learn the model parameter w, so that for a new example x, we can classify x to be class y^* if $y^* = \arg\max_y f(x, y)$.

In analogy to classical SVMs, we would like to train w from labeled examples D by solving the following optimization problem:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}||w||^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & f_w(x_i, y) - f_w(x_i, y_i) \leq \xi_i - 1, \quad \forall i, \quad \forall y \neq y_i \\ & \xi_i \geq 0, \qquad \forall i \end{aligned} \tag{6.3}$$

where C is the trade-off parameter similar to that in SVMs, and ξ_i is the slack variable for the i-the training example to handle the case of soft margin.

The optimization problem in (6.3) is equivalent to the following optimization problem:

$$\begin{aligned}
&\min_{w,\xi} \quad \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
&\text{s.t.} \quad \max_h w^\top \Phi(x_i,h,y) - \max_{h'} w^\top \Phi(x_i,h',y_i) \\
&\qquad \leq \xi_i - \delta(y,y_i), \quad \forall i, \quad \forall y \\
&\text{where} \quad \delta(y,y_i) = \begin{cases} 1 & \text{if } y \neq y_i \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{6.4}$$

6.2.2 Semi-Convexity and Primal Optimization

Similar to LSVMs, MMHCRFs have the property of semi-convexity. Note that $f_w(x,y)$ is a maximum of a set of functions, each of which is linear in w, so $f_w(x)$ is convex in w. If we restrict the domain of h' in (6.4) to a single choice, the optimization problem of (6.4) becomes convex [16]. This is in analog to restricting the domain of the latent variables for the positive examples to a single choice in LSVMs [29]. But here we are dealing with multi-class classification, our "positive examples" are those $\langle x_i, y\rangle$ pairs where $y = y_i$.

We can compute a local optimum of (6.4) using a coordinate descent algorithm similar to LSVMs [29]:

1. Holding w, ξ fixed, optimize the latent variables h' for the $\langle x_i, y_i\rangle$ pair:

$$h_{i,y_i} = \arg\max_{h'} w^\top \Phi(x_i,h',y_i)$$

2. Holding h_{i,y_i} fixed, optimize w, ξ by solving the following optimization problem:

$$\begin{aligned}
&\min_{w,\xi} \quad \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
&\text{s.t.} \quad \max_h w^\top \Phi(x_i,h,y) - w^\top \Phi(x_i,h_{i,y_i},y_i) \\
&\qquad \leq \xi_i - \delta(y,y_i), \quad \forall i, \quad \forall y
\end{aligned} \tag{6.5}$$

It can be shown that both steps always improve or maintain the objective [29].

The optimization problem in Step 1 can be solved efficiently for certain structures of h' (see Sec. 6.2.4 for details). The optimization problem in Step 2 involves solving a quadratic program (QP) with piecewise linear constraints. Although it is possible to solve it directly using barrier methods [16], we will not be able to take advantage of existing highly optimized solvers (e.g., CPLEX) which only accept linear constraints. It is desirable to convert (6.5) into a standard quadratic program with only linear constraints.

One possible way to convert (6.5) into a standard QP is to solve the following convex optimization problem:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & w^\top \Phi(x_i, h, y) - w^\top \Phi(x_i, h_{i,y_i}, y_i) \\ & \leq \xi_i - \delta(y, y_i), \qquad \forall i, \forall h, \forall y \end{aligned} \tag{6.6}$$

It is easy to see that (6.5) and (6.6) are equivalent, and all the constraints in (6.6) are linear. Unfortunately, the optimization problem in (6.6) involves an exponential number of constraints – for each example x_i and each possible labeling y, there are exponentially many possible h's.

We would like to perform optimization over a much smaller set of constraints. One solution is to use a cutting plane algorithm similar to that used in structured SVMs [3] and CRFs [101]. In a nutshell, the algorithm starts with no constraints (which corresponds to a relaxed version of (6.6)), then iteratively finds the "most violated" constraints and adds those constraints. It can be shown that this algorithm computes arbitrarily close approximation to the original problem of (6.6) by evaluating only a polynomial number of constraints.

More importantly, the optimization problem in (6.6) has certain properties that allow us to find and add constraints in an efficient way. For a fixed example x_i and a possible label y, define $h_{i,y}$ as follows:

$$h_{i,y} = \arg\max_h w^\top \Phi(x_i, h, y)$$

Consider the following two set of constraints for the $\langle x_i, y \rangle$ pair:

$$\begin{aligned} & w^\top \Phi(x_i, h_{i,y}, y) - w^\top \Phi(x_i, h_{i,y_i}, y_i) \\ & \leq \xi_i - \delta(y, y_i) \end{aligned} \tag{6.7}$$

$$\begin{aligned} & w^\top \Phi(x_i, h, y) - w^\top \Phi(x_i, h_{i,y_i}, y_i) \\ & \leq \xi_i - \delta(y, y_i), \quad \forall h \end{aligned} \tag{6.8}$$

It is easy to see that (6.7) and (6.8) define the same set of constraints, i.e., (6.7) implies (6.8) and vice versa. This suggests that for a fixed $\langle x_i, y \rangle$ pair, we only need to consider the constraint involving $h_{i,y}$.

Putting everything together, we learn the model parameter w by iterating the following two steps.

1. Fixing w, ξ, optimize the latent variable h for each pair $\langle x_i, y \rangle$ of an example x_i and a possible labeling y:
$$h_{i,y} = \arg\max_h w^\top \Phi(x_i, h, y)$$

2. Fixing $h_{i,y} \quad \forall i, \quad \forall y$, optimize w, ξ by solving the following optimization problem:
$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & w^\top \Phi(x_i, h_{i,y}, y) - w^\top \Phi(x_i, h_{i,y_i}, y_i) \\ & \le \xi_i - \delta(y, y_i), \quad \forall i, \quad \forall y \end{aligned} \tag{6.9}$$

Step 1 in the above algorithm can be efficiently solved for certain structured h (Sec. 6.2.4). Step 2 involves solving a quadratic program with $N \times |\mathcal{Y}|$ constraints.

The optimization in (6.9) is very similar to the primal problem of a standard multi-class SVM [19]. In fact, if $h_{i,y}$ is the same for different y's, it is just a standard SVM and we can use an off-the-shelf SVM solver to optimize (6.9). Unfortunately, the fact that $h_{i,y}$ can vary with different y's means that we cannot directly use standard SVM packages. We instead develop our own optimization algorithm.

6.2.3 Dual Optimization

In analog to classical SVMs, it is helpful to solve the problem in (6.9) by examining its dual. To simplify the notation, let us define $\Psi(x_i, y) = \Phi(x_i, h_{i,y}, y) - \Phi(x_i, h_{i,y_i}, y_i)$. Then the dual problem of (6.9) can be written as follows:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \sum_y \alpha_{i,y} \delta(y, y_i) - \frac{1}{2}||\sum_i \sum_y \alpha_{i,y} \Psi(x_i, y)||^2 \\ \text{s.t.} \quad & \sum_y \alpha_{i,y} = C, \quad \forall i \\ & \alpha_{i,y} \ge 0, \quad \forall i, \quad \forall y \end{aligned} \tag{6.10}$$

The primal variable w can be obtained from the dual variables α as follows:

$$w = -\sum_i \sum_y \alpha_{i,y} \Psi(x_i, y)$$

Note that (6.10) is quite similar to the dual form of standard multi-class SVMs. In fact, if $h_{i,y}$ is a deterministic function of x_i, (6.10) is just a standard dual form of SVMs.

Similar to classical SVMs, we can also obtain a kernelized version of the algorithm by defining a kernel function of size $N \times |\mathcal{Y}|$ by $N \times |\mathcal{Y}|$ in the following form:

$$K(i, y; j, y') = \Psi(x_i, y)^\top \Psi(x_j, y')$$

Let us define α as the concatenation of $\{\alpha_{i,y} : \forall i \ \ \forall y\}$, so the length of α is $N \times |\mathcal{Y}|$. Define Δ as a vector of the same length. The (i, y)-th entry of Δ is 1 if $y \neq y_i$, and 0 otherwise. Then (6.10) can be written as:

$$\begin{aligned} \max_\alpha \quad & \alpha^\top \Delta - \frac{1}{2}\alpha^\top K \alpha \\ \text{s.t.} \quad & \sum_y \alpha_{i,y} = C, \qquad \forall i \\ & \alpha_{i,y} \geq 0, \qquad \forall i, \forall y \end{aligned} \tag{6.11}$$

Note the matrix K in (6.11) only depends on the dot-product between feature vectors of different $\langle x_i, y \rangle$ pairs. So our model has a very intuitive and interesting interpretation – it defines a particular kernel function that respects the latent structures.

It is easy to show that the optimization problem in (6.10) is concave, so we can find its global optimum. But the number of variables is $N \times |\mathcal{Y}|$, where N is the number of training examples, and $|\mathcal{Y}|$ is the size of all possible class labels. So it is infeasible to use a generic QP solver to optimize it.

Instead, we decompose the optimization problem of (6.10) and solve a series of smaller QPs. This is similar to the sequential minimal optimization (SMO) used in SVM [19, 73] and M^{3}N [102]. The basic idea of this algorithm is to choose all the $\{\alpha_{i,y} : \forall y \in \mathcal{Y}\}$ for a particular training example x_i and fix all the other variables $\{\alpha_{k,y'} : \forall k : k \neq i, \forall y' \in \mathcal{Y}\}$ that do not involve x_i. Then instead of solving a QP involving all the variables $\{\alpha_{i,y} : \forall i, \forall y\}$, we can solve a much smaller QP only involving $\{\alpha_{i,y} : \forall y\}$. The number of variables of this smaller QP is $|\mathcal{Y}|$, which is much smaller than $N \times |\mathcal{Y}|$.

First we write the objective of (6.10) in terms of $\{\alpha_{i,y} : \forall y\}$ as follows:

$$\begin{aligned}
&\mathcal{L}(\{\alpha_{i,y} : \forall y\}) \\
=& \sum_y \alpha_{i,y}\delta(y, y_i) - \frac{1}{2}\Bigg[|| \sum_y \alpha_{i,y}\Psi(x_i, y)||^2 \\
&+2\Big(\sum_y \alpha_{i,y}\Psi(x_i, y)\Big)^\top \Big(\sum_{k:k\neq i}\sum_{y'} \alpha_{k,y'}\Psi(x_k, y')\Big)\Bigg] \\
&+\text{other terms not involving } \{\alpha_{i,y} : \forall y\}
\end{aligned}$$

The smaller QP corresponding to $\langle x_i, y_i\rangle$ can be written as follows:

$$\begin{aligned}
\max_{\alpha_{i,y}:\forall y} \quad & \mathcal{L}(\{\alpha_{i,y} : \forall y\}) \\
\text{s.t.} \quad & \sum_y \alpha_{i,y} = C \\
& \alpha_{i,y} \geq 0, \quad \forall y
\end{aligned} \tag{6.12}$$

Note $\sum_{k:k\neq i}\sum_{y'} \alpha_{k,y'}\Psi(x_k, y')$ can be written as:

$$-w - \sum_y \alpha_{i,y}\Psi(x_i, y)$$

So as long as we maintain (and keep updating) the global parameter w and keep track of $\alpha_{i,y}$ and $\Psi(x_i, y)$ for each example $\langle x_i, y_i\rangle$, we do not need to actually do the summation $\sum_{k:k\neq i}\sum_{y'}$ when optimizing (6.12). In addition, when we solve the QP involving $\alpha_{i,y}$ for a fixed i, all the other constraints involving $\alpha_{k,y}$ where $k \neq i$ are not affected. This is not the case if we try to optimize the primal problem in (6.9). If we try to optimize the primal variable w by only considering the constraints involving the i-th examples, it is possible that the new w obtained from the optimization might violate the constraints imposed by other examples. There is also work [18] showing that the dual optimization has a better convergence rate.

6.2.4 Finding the Optimal h

The alternating coordinate descent algorithm for learning the model parameter w described in Sec. 6.2.2 assumes we have an inference algorithm for finding the optimal h^* for a fixed $\langle x, y\rangle$ pair:

$$h^* = \arg\max_h w^\top \Phi(x, h, y) \tag{6.13}$$

In order to adopt our approach to problems involving different latent structures, this is the only component of the algorithm that needs to be changed.

If $h = (z_1, z_2, ..., z_m)$ forms a tree-structured graphical model, the inference problem in (6.13) can be solved exactly, e.g., using the Viterbi dynamic programming algorithm for trees. We can also solve it using standard linear programming as follows [103, 108]. We introduce variables $\mu_{j\mathfrak{a}}$ to denote the indicator $\mathbb{1}(z_j = \mathfrak{a})$ for all vertices $j \in \mathcal{V}$ and their values $\mathfrak{a} \in \mathcal{H}$. Similarly, we introduce variables $\mu_{jk\mathfrak{ab}}$ to denote the indicator $\mathbb{1}(z_j = \mathfrak{a}, z_k = \mathfrak{b})$ for all edges $(j,k) \in \mathcal{E}$ and the values of their nodes, $\mathfrak{a} \in \mathcal{H}$, $\mathfrak{b} \in \mathcal{H}$. We use $\psi_j(z_j)$ to collectively represent the summation of all the unary potential functions in (6.1) that involve the node $j \in \mathcal{V}$. We use $\psi_{jk}(z_j, z_k)$ to collectively represent the summation of all the pairwise potential functions in (6.1) that involve the edge $jk \in \mathcal{E}$. The problem of finding of optimal h^* can be formulated into the following linear programming (LP) problem:

$$\begin{aligned}
\max_{0 \leq \mu \leq 1} & \sum_{j \in \mathcal{V}} \sum_{\mathfrak{a} \in \mathcal{H}} \mu_{j\mathfrak{a}} \psi_j(\mathfrak{a}) + \sum_{jk \in \mathcal{E}} \sum_{\mathfrak{a} \in \mathcal{H}} \sum_{\mathfrak{b} \in \mathcal{H}} \mu_{jk\mathfrak{ab}} \psi_{jk}(\mathfrak{a}, \mathfrak{b}) \\
\text{s.t.} \quad & \sum_{\mathfrak{a} \in \mathcal{H}} \mu_{j\mathfrak{a}} = 1, \quad \forall j \in \mathcal{V} \\
& \sum_{\mathfrak{a} \in \mathcal{H}} \sum_{\mathfrak{b} \in \mathcal{H}} \mu_{jk\mathfrak{ab}} = 1, \quad \forall jk \in \mathcal{E} \\
& \sum_{\mathfrak{a} \in \mathcal{H}} \mu_{jk\mathfrak{ab}} = \mu_{k\mathfrak{b}}, \quad \forall jk \in \mathcal{E}, \quad \forall \mathfrak{b} \in \mathcal{H} \\
& \sum_{\mathfrak{b} \in \mathcal{H}} \mu_{jk\mathfrak{ab}} = \mu_{j\mathfrak{a}}, \quad \forall jk \in \mathcal{E}, \quad \forall \mathfrak{a} \in \mathcal{H}
\end{aligned} \tag{6.14}$$

If the optimal solution of this LP is integral, we can recover h^* from μ^* very easily. It has been shown that if $\mathcal{E}$ forms a forest, the optimal solution of this LP is guaranteed to be integral [103, 108]. For general graph topology, the optimal solution of this LP can be fractional, which is not surprising, since the problem in (6.13) is NP-hard for general graphs. Although the LP formulation does not seem to be particularly advantageous in the case of tree-structured models, since they can be solved by Viterbi dynamic programming anyway, the LP formulation provides a more general way of approaching other structures (e.g., Markov networks with sub-modular potentials, matching [103]).

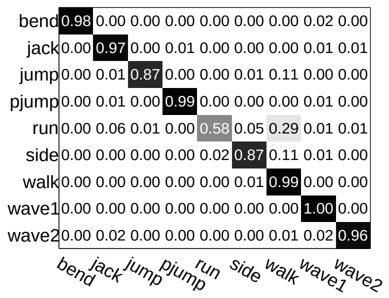
6.2.5 Comparison with HCRF

MMHCRFs are closely related to HCRFs. The main difference lies in their different learning criteria – maximizing the margin in MMHCRFs, and maximizing the conditional likelihood in HCRFs. As a result, the learning algorithms for MMHCRFs and HCRFs involve solving two different types of inference problems – maximizing over h, versus summing over h.

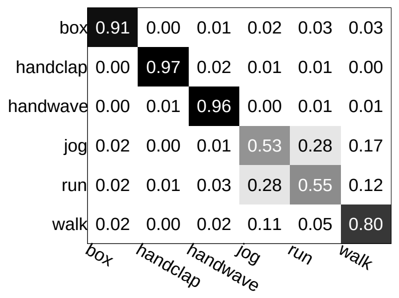
From the computational perspective, if h has a tree structure and $|\mathcal{H}|$ is relatively small, both inference problems (max vs. sum) can be solved exactly, using dynamic programming and belief propagation, respectively. But the inference problem (maximization) in MMHCRFs can deal with a much wider range of latent structures. Here are a few examples [103] (although these problems are not addressed in this dissertation):

- Binary Markov networks with sub-modular potentials, commonly encountered in figure/ground segmentation [49]. MMHCRFs can use LP [103] or graph-cut [49] to solve this problem. For HCRFs, this problem can only be solved approximately, e.g., using loopy BP or mean-field variational methods.
- Matching/correspondence. This structure can be solved by MMHCRFs using LP [89, 103]. It is not clear how HCRFs can be used in this situation, since it requires summing over all the possible matchings.
- Tree-structures, but each node in h can have a large number of possible labels (e.g., all the possible pixel locations in an image), i.e., $|\mathcal{H}|$ is big. If the pairwise potentials have certain properties, distance transform [30] can be applied in MMHCRFs to solve the problem. This is essentially what has been done in [29]. This problem can also be solved by HCRFs using convolution. But distance transform ($O(|\mathcal{H}|)$) is more efficient than convolution ($O(|\mathcal{H}| \log |\mathcal{H}|)$).

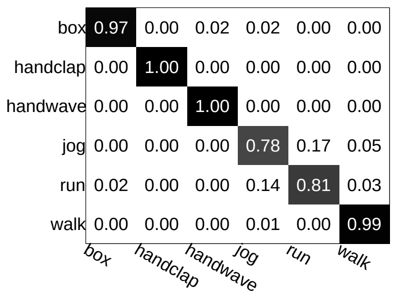
From the modeling perspective, we believe MMHCRFs are better suited for classification than HCRFs. This is because HCRFs require summing over exponentially many h's. In order to maximize the conditional likelihoods, the learning algorithm of HCRFs has to try very hard to push the probabilities of many "wrong" labelings of h's to be close to zero. But in MMHCRFs, the learning algorithm only needs to push apart the "correct" labeling and its next best competitor. Conceptually, the modeling criterion of MMHCRFs is easier to achieve than that of HCRFs. A more rigorous analysis of these two different learning criteria will be an interesting and important future work.



(a) Weizmann: per-frame



(b) KTH: per-frame



(c) KTH: per-video

Figure 6.2: Confusion matrices of classification results on Weizmann and KTH dataset. The confusion matrix of per-video classification on the Weizmann dataset is not shown, since it is simply a perfect diagonal matrix.

6.3 Experiments

We test our algorithm on the Weizmann human action dataset [9] and the KTH human motion dataset [90]. Performance on these benchmarks is saturating – state-of-the-art approaches achieve near-perfect results. We show our method outperforms the HCRF model [113] and is comparable to other state-of-the-art approaches. In order to do a fair comparison with [113], we use the same split of training/testing data. We also use the same patch initialization technique in [113] to find ten salient patches in each image. We set $C = 1$ in all the experiments.

Weizmann dataset: Similar to the experiments in Chapter 5, we learn our model with different sizes of possible part labels, $|\mathcal{H}| = 6, 10, 20$. During testing, our model performs per-frame classification, then obtains the class label for the video by majority voting.

The comparison with the HCRF model [113] is shown in Table 6.1. Our model outperforms HCRF in all three cases. The comparison with other approaches is summarized in Table 6.2. We should emphasize that we do not attempt a direct comparison, since Niebles & Fei-Fei [69] does not require stabilization, and Blank et.al [9] uses a different measurement (per-cube accuracy). We show the confusion matrix of our approach with $|\mathcal{H}| = 10$ for per-frame classification in Fig. 6.2(a). The confusion matrix for per-video classification is omitted, since it is just a perfect diagonal matrix.

KTH dataset: Similarly, we learn the model with $|\mathcal{H}| = 6, 10, 20$ on the KTH dataset. The confusion matrices of our approach (with $|\mathcal{H}| = 10$) for both per-frame and per-video

method	$\|\mathcal{H}\|$=6	$\|\mathcal{H}\|$=10	$\|\mathcal{H}\|$=20
HCRF	0.8682 0.9167	0.9029 0.9722	0.8557 0.9444
Our approach	**0.8996** **0.9722**	**0.9311** **1.0000**	**0.8891** **0.9722**

Table 6.1: Comparison of our approach with the HCRF model [113] on the Weizmann dataset. The first number in each cell is the accuracy of per-frame classification. The second number is the accuracy of per-video classification.

method	per-frame	per-video	per-cube
Our method	**0.9311**	**1.0000**	N/A
Wang & Mori [113]	0.9029	0.9722	N/A
Jhuang et al. [45]	N/A	0.9880	N/A
Niebles & Fei-Fei [69]	0.5500	0.7280	N/A
Blank et al. [9]	N/A	N/A	0.9964

Table 6.2: Comparison of classification accuracy with previous work on the Weizmann dataset.

classification are shown in Fig. 6.2(b,c). The comparison with the HCRF model is summarized in Table 6.3. Again, our approach outperforms the HCRF model.

The comparison with other approaches is summarized in Table 6.4. Again, we would like to emphasize that this is not a direct comparison, due to all sorts of variations of the experiment setups in different methods listed in Table 6.4. We only show the results to demonstrate that our approach is comparable to other state-of-the-art methods.

method	$\|\mathcal{H}\|$=6	$\|\mathcal{H}\|$=10	$\|\mathcal{H}\|$=20
HCRF	0.6633 0.7855	0.6698 0.8760	0.6444 0.7512
Our approach	**0.7064** **0.8475**	**0.7853** **0.9251**	**0.7486** **0.8966**

Table 6.3: Comparison of our approach with the HCRF model on the KTH dataset. The first number in each cell is the accuracy of per-frame classification. The second number is the accuracy of per-video classification.

method	**accuracy**
Our method	**0.9251**
Liu & Shah [60]	**0.9416**
Jhuang et al. [45]	0.9170
Wang & Mori [113]	0.8760
Niebles et al. [70]	0.8150
Dollár et al. [23]	0.8117
Schuldt et al. [90]	0.7172

Table 6.4: Comparison of per-video classification accuracy with previous approaches on the KTH dataset.

6.4 Summary

We have presented the max-margin hidden conditional random field – a learning framework for training multi-class classifiers with structured latent variables. We propose an efficient learning algorithm based on the cutting plane method and decomposed dual optimization. Our proposed model is quite general and can be applied in a wide range of latent structures, as long as we have an algorithm for solving the inference in Sec. 6.2.4. For a lot of vision applications, this inference is easier than computing the summation over all the latent variables, which is required by HCRF training. Our experimental results show that the max-margin learning achieves better performance than the conditional likelihood learning, which is consistent with similar findings in the structured output learning literature [3, 102, 103].

As future work, we would like to apply our model in various vision problems that involve latent structures, and to derive efficient approximation algorithms for general graph structures, of which exact inference is intractable. We also plan to further investigate the theoretical differences between the max-margin and the conditional likelihood learning criteria.

Chapter 7

Conclusion and Future Work

In this dissertation, I have presented a series of models and algorithms for solving two important problems in the general area of "looking at people", namely, *human pose estimation* and *human action recognition*. A common theme of the proposed approaches is that they all involve learning *structured models*. In particular, I have proposed methods for modeling structures outputs for human pose estimation, structured latent variables for part-based action recognition, and temporal structures for sequence-based action recognition.

The line of work presented in this dissertation leads to many interesting and exciting directions for future research. In the following, I will briefly highlight several directions for future research.

Structured Output-Structured Hidden Variables: The first obvious direction for future work is to consider problems involving more complex structures, i.e., both the outputs y and hidden variables h are structured. Many problems in computer vision can be formulated using this setting. Consider the following example. Suppose we have a collection of images x with associated tags y. Let us assume y have certain structures. One particular form of such structures is a MRF model where the edges are formed by examining the co-occurrence statistics of pairs of tags in a corpus [75] – if two tags appear together frequently (e.g., "car" and "road"), an edge in the MRF models the correlation of these two tags. If an image x is segmented into several regions, we can assume that each tag is associated with one or more regions in the image. Of course, the correspondence between tags and image regions are not observed in the training data, so we can consider this information as hidden variables. If we can successfully learn a model that combines all these inputs (images), outputs (tags) and hidden variables (correspondence between tags and images regions), we can use the model for a lot applications, e.g., auto-tagging images, image retrieval, etc. There are many other exciting applications in computer vision that can be formulated into this setting.

Weakly Supervised Learning: We can extend the framework to the case where the hidden variable h is observed in some of the training data. This line of research is related to our previous work [37] on learning with incomplete information. The hope is that observing h on some training data will make it easier to train the model and to learn semantically meaningful hidden structures.

Multiple Tasks: Thirdly, instead of having one layer of hidden variable h and one layer of output y, we can extend to more complicated models involving multiple layers of h and y. Different layers of y (multi-class or structured-output) correspond to different high-level tasks. Different layers of h correspond to different aspects of the assumptions, e.g., in the object recognition case, we can have one layer of hidden variables h_1 for the figure/ground labeling, another layer of hidden variables h_2 for the object bounding box. h_1 and h_2 can both interact with each other. The hope is that different tasks and different hidden variables will reinforce each other and provide different views of the data.

I believe *structured models* open the door to many exciting research problems. They will allow us to explore vast amount of visual data in which rich structures are ubiquitous.

Bibliography

[1] Ankur Agarwal and Bill Triggs. A local basis representation for estimating human pose from cluttered images. In *Asian Conference on Computer Vision*, 2006.

[2] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[3] Yasemin Altun, Thomas Hofmann, and Ioannis Tsochantaridis. SVM learning for interdependent and structured output spaces. In G. Bakir, T. Hofman, B. Scholkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, editors, *Machine Learning with Structured Outputs*. MIT Press, 2006.

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[5] Serge Belongie, Greg Mori, and Jitendra Malik. Matching with shape contexts. In *Analysis and Statistics of Shapes*. Birkhäuser, 2005.

[6] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] Alessandro Bissacco, Ming-Hsuan Yang, and Stefano Soatto. Detecting humans via their pose. In *Advances in Neural Information Processing Systems*, volume 19, pages 169–176. MIT Press, 2007.

[9] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *IEEE International Conference on Computer Vision*, 2005.

[10] David M. Blei and John D. Lafferty. Correlated topic models. In *Advances in Neural Information Processing Systems*, volume 18, 2006.

[11] David M. Blei and Jon D. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2008.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

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

[14] Aaron F. Bobick and Andrew 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, December 1997.

[15] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Scene classification via pLSA. In *European Conference on Computer Vision*, volume 4, pages 517–530, 2006.

[16] Stephen Boyd and Lieven Vanderghe. *Convex Optimization*. Cambridge University Press, 2004.

[17] M. Brand. Shadow puppetry. In *IEEE International Conference on Computer Vision*, 1999.

[18] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1757–1774, August 2008.

[19] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[20] David Crandell, Pedro F. Felzenszwalb, and Daniel P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 10–17, 2005.

[21] Ross Cutler and Larry 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.

[22] Navneet Dalal and Bill Triggs. Histogram of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[23] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV'05 Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[24] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, 2003.

[25] Claudio Fanti, Lihi Zelnik-Manor, and Pietro Perona. Hybrid models for human motion recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[26] Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recongition*, 2008.

[27] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006.

[28] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531, 2005.

[29] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[30] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, January 2005.

[31] Xiaolin Feng and Pietro Perona. Human action recognition by sequence of movelet codewords. In *International Symposium on 3D Data Processing Visualization and Transmission*, 2002.

[32] Rob Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning object categories from google's image search. In *IEEE International Conference on Computer Vision*, volume 2, pages 1816–1823, 2005.

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

[34] Patrick Flaherty, Guri Giaever, Jochen Kumm, Michael I. Jordan, and Adam P. Arkin. A latent variable model for chemogenomic profiling. *Bioinformatics*, 21(15):3286–3293, 2005.

[35] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*, volume 2, pages 1458–1465, 2005.

[36] R. Gross and J. Shi. The cmu motion of body(mobo) database. Technical Report CMU-RI-TR-01-18, CMU, 2001.

[37] Gholamreza Haffari, Yang Wang, Shaojun Wang, Greg Mori, and Feng Jiao. Boosting with incomplete information. In *The 25th International Conference on Machine Learning*, 2008.

[38] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of Twenty-Second Annual International Conference on Research and Development in Information Retrieval(SIGIR)*, pages 50–57, 1999.

[39] Gang Hua, Ming-Hsuan Yang, and Ying Wu. Learning to estimate human pose with data driven belief propagation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 747–754, 2005.

[40] Jonathan Huang and Thomas Malisiewicz. Fitting a hierarchical logistic normal distribution. http://www.cs.cmu.edu/∼jch1/research/hln/hlnfit.html.

[41] Nazli Ikizler, R. Gokberk Cinbis, Selen Pehlivan, and Pinar Duygulu. Recognizing actions from still images. In *International Conference on Pattern Recognition*, 2008.

[42] Nazli Ikizler and David Forsyth. Searching video for complex activities with finite state models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[43] S. Ioffe and D. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision*, 43(1):45–68, 2001.

[44] Sergey Ioffe and David Forsyth. Finding people by sampling. In *IEEE International Conference on Computer Vision*, volume 2, pages 1092–1097, 1999.

[45] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *IEEE International Conference on Computer Vision*, 2007.

[46] Shanon X. Ju, Michael J. Black, and Yaser Yaccob. Cardboard people: A parameterized model of articulated image motion. In *International Conference on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

[47] Yan Ke, Rahul Sukthankar, and Martial Hebert. Efficient visual event detection using volumetric features. In *IEEE International Conference on Computer Vision*, volume 1, pages 166–173, 2005.

[48] Yan Ke, Rahul Sukthankar, and Martial Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision*, 2007.

[49] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004.

[50] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289, 2001.

[51] Xiangyang Lan and Daniel P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *IEEE International Conference on Computer Vision*, volume 1, pages 470–477, 2005.

[52] Ivan Laptev and T. Lindeberg. Space-time interest points. In *IEEE International Conference on Computer Vision*, 2003.

[53] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[54] Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *IEEE International Conference on Computer Vision*, 2007.

[55] Ben Laxton, Jongwoo Lim, and David Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[56] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A maximum entropy framework for part-based texture and object recognition. In *IEEE International Conference on Computer Vision*, volume 1, pages 832–838, 2005.

[57] Y. LeCun, S. Schopra, R. Radsell, R. Marc'Aurelio, and F. Huang. A tutorial on energy-based learning. In T. Hofman, B. Scholkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[58] Mun Wai Lee and Isaac Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 334–341, 2004.

[59] James L. Little and Jeffery E. Boyd. Recognizing people by their gait: The shape of motion. *Videre*, 1(2):1–32, 1998.

[60] Jingren Liu and Mubarak Shah. Learning human actions via information maximization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recongition*, 2008.

[61] Wei-Lwun Lu, Kenji Okuma, and James J. Little. Tracking and recognizing actions of multiple hockey players using the boosted particle filter. *Image and Vision Computing*, 27(1-2):189–205, January 2009.

[62] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[63] Marina Meila and Michael I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

[64] Thomas P. Minka. Estimating a Dirichlet distribution. Technical report, Massachusetts Institute of Technology, 2000.

[65] Greg Mori. Guiding model search using segmentation. In *IEEE International Conference on Computer Vision*, volume 2, pages 1417–1423, 2005.

[66] Greg Mori, Serge Belongie, and Jitendra Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005.

[67] Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision*, volume 3, pages 666–680, 2002.

[68] Greg Mori, Xiaofeng Ren, Alyosha Efros, and Jitendra Malik. Recovering human body configuration: Combining segmentation and recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 326–333, 2004.

[69] Juan Carlos Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[70] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *British Machine Vision Conference*, volume 3, pages 1249–1258, 2006.

[71] Sebastian Nowozin, Gokhan Bakir, and Koji Tsuda. Discriminative subsequence mining for action classification. In *IEEE International Conference on Computer Vision*, 2007.

[72] Nuria Olivera, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96(2):163–180, November 2004.

[73] John C. Platt. Using analytic QP and sparseness to speed training of support vector machines. In *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999.

[74] Ramprasad Polana and Randal C. Nelson. Detection and recognition of periodic, non-rigid motion. *International Journal of Computer Vision*, 23(3):261–282, June 1997.

[75] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *ACM Multimedia*, 2007.

[76] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, Cambridge, MA, 2005.

[77] Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1848–1852, June 2007.

[78] Deva Ramanan. Learning to parse images of articulated bodies. In *Advances in Neural Information Processing Systems*, volume 19, pages 1129–1136, 2007.

[79] Deva Ramanan and David A. Forsyth. Automatic annotation of everyday movements. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[80] Deva Ramanan and David A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[81] Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Strike a pose: Tracking people by finding stylized poses. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 271–278, 2005.

[82] Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, January 2007.

[83] Deva Ramanan and Cristian Sminchisescu. Training deformable models for localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 206–213, 2006.

[84] Xiaofeng Ren, Alexander Berg, and Jitendra Malik. Recovering human body configurations using pairwise constraints between parts. In *IEEE International Conference on Computer Vision*, volume 1, pages 824–831, 2005.

[85] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action MACH: A spatial-temporal maximum average correlation height filter for action recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[86] R. Rosales and S. Sclaroff. Learning body pose via specialized maps. In *Advances in Neural Information Processing Systems*. MIT Press, 2001.

[87] Bryan C. Russell, Alexei A. Efros, Josef Sivic, William T. Freeman, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[88] Konrad Schindler and Luc Van Gool. Action snippets: How many frames does action recognition require? In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[89] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, Berlin, 2003.

[90] Christian Schuldt, Lvan Laptev, and Barbara Caputo. Recognizing human actions: a local SVM approach. In *IEEE International Conference on Pattern Recognition*, volume 3, pages 32–36, 2004.

[91] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *IEEE International Conference on Computer Vision*, volume 2, pages 750–757, 2003.

[92] Eli Shechtman and Michal Irani. Space-time behavior based correlation. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

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

[94] Leonid Sigal and Michael J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2041–2048, 2006.

[95] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their location in images. In *IEEE International Conference on Computer Vision*, volume 1, pages 370–377, 2005.

[96] Cristian Sminchisescu, Atul Kanaujia, Zhiguo Li, and Dimitris Metaxas. Conditional models for contextual human motion recognition. In *IEEE International Conference on Computer Vision*, 2005.

[97] Yang Song, Luis Goncalves, and Pietro Perona. Unsupervised learning of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):814–827, July 2003.

[98] Praveen Srinivasan and Jianbo Shi. Bottom-up recognition and parsing of the human body. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[99] Erik B. Sudderth, Michael I. Mandel, William T. Freeman, and Alan S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *Advances in Neural Information Processing Systems*, pages 1369–1376. MIT Press, 2004.

[100] Josephine Sullivan and Stefan Carlsson. Recognizing and tracking human action. In *European Conference on Computer Vision LNCS 2352*, volume 1, pages 629–644, 2002.

[101] Martin Szummer, Pushmeet Kohli, and Derek Hoiem. Learning CRFs using graph cuts. In *European Conference on Computer Vision*, 2008.

[102] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[103] Ben Taskar, Simon Lacoste-Julien, and Michael I. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006.

[104] Christian Thuran and Václav Hlaváč. Pose primitive based human action recognition in videos or still images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[105] K. Toyama and A. Blake. Probabilistic exemplar-based tracking in a metric space. In *IEEE International Conference on Computer Vision*, volume 2, pages 50–57, 2001.

[106] Duan Tran and David Forsyth. Configuration estimates improve pedestrian finding. In *Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2008.

[107] Tran The Truyen, Dinh Q. Phung, Hung H. Bui, and Svetha Venkatesh. AdaBoost.MRF: Boosted markov random forests and application to multilevel activity recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1686–1693, 2006.

[108] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, November 2005.

[109] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, July 2005.

[110] Sy Bor Wang, Ariadna Quattoni, Louis-Philippe Morency, David Demirdjian, and Trevor Darrell. Hidden conditional random fields for gesture recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[111] Yang Wang and Greg Mori. Boosted multiple deformable trees for parsing human poses. In *2nd Workshop on Human Motion Understanding, Modeling, Capture and Animation*, 2007.

[112] Yang Wang and Greg Mori. Multiple tree models for occlusion and spatial constraints in human pose estimation. In *European Conference on Computer Vision*, 2008.

[113] Yang Wang and Greg Mori. Learning a discriminative hidden part model for human action recognition. In *Advances in Neural Information Processing Systems*, volume 21. MIT Press, 2009.

[114] Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[115] Yang Wang and Greg Mori. Human action recognition by semi-latent topic models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear.

[116] Yang Wang, Payam Sabzmeydani, and Greg Mori. Semi-latent Dirichlet allocation: A hierarchical model for human action recognition. In *The 2nd Workshop on Human Motion Understanding, Modeling, Capture and Animation*, 2007.

[117] Shu-Fai Wong, Tae-Kyun Kim, and R. Cipolla. Learning motion categories using both semantic and structure information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recongition*, 2007.

[118] Tao Xiang and Shaogang Gong. Beyond tracking: Modelling activity and understanding behaviour. *International Journal of Computer Vision*, 67(1):21–51, 2006.

[119] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992.

[120] Alper Yilmaz and Mubarak Shah. Recognizing human actions in videos acquired by uncalibrated moving cameras. In *IEEE International Conference on Computer Vision*, 2005.

[121] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.