Efficient Human Pose Estimation with Image-dependent Interactions

Benjamin John Sapp

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2012

Ben Taskar, Associate Professor
Computer and Information Science
Supervisor of Dissertation

Jianbo Shi, Associate Professor
Computer and Information Science
Graduate Group Chairperson

Dissertation Committee

Kostas Daniilidis, Professor
Computer and Information Science
University of Pennsylvania

Jianbo Shi, Associate Professor
Computer and Information Science
University of Pennsylvania

Camillo J. Taylor, Associate Professor
Computer and Information Science
University of Pennsylvania

David Forsyth, Professor
Computer Science
University of Illinois

Efficient Human Pose Estimation with Image-dependent Interactions

Acknowledgments

I'd like to thank . . .

ABSTRACT

Efficient Human Pose Estimation with Image-dependent Interactions

Benjamin John Sapp

Ben Taskar

Human pose estimation from monocular images is one of the most challenging and computationally demanding problems in computer vision. Standard models such as Pictorial Structures consider interactions between kinematically connected joints or limbs, leading to inference cost that is quadratic in the number of pixels. As a result, researchers and practitioners have restricted themselves to simple models which only measure the quality of limb-pair possibilities by their 2D geometric plausibility.

In this talk, we propose novel methods which allow for efficient inference in richer models with data-dependent interactions. First, we introduce structured prediction cascades, a structured analog of binary cascaded classifiers, which learn to focus computational effort where it is needed, filtering out many states cheaply while ensuring the correct output is unfiltered. Second, we propose a way to decompose models of human pose with cyclic dependencies into a collection of tree models, and provide novel methods to impose model agreement.

These techniques allow for sparse and efficient inference on the order of minutes per image or video clip. As a result, we can afford to model pairwise interaction potentials much more richly with data-dependent features such as contour continuity, segmentation alignment, color consistency, optical flow and more. We show empirically that these richer models are worthwhile, obtaining significantly more accurate pose estimation on popular datasets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Geman quote"

Why pose estimation?

The idea of an intelligent robot performing a variety of mundane and extraordinary tasks, up to and exceeding human performance, has captured the hearts and minds of people since at least The Renaissance. A key feature of much of this romantic vision is that robots can interact with *us* - working with, around and for humans. An understanding of human pose is a crucial component to making this compelling dream becoming a reality.

In the more practical and not-too-distant future, understanding human pose from images has enormous potential to help in many computer vision tasks: semantic indexing of images and video (Ferrari et al., 2009), action recognition (Tran et al., 2011), human-object interaction (Yao and Fei-Fei, 2012), scene understanding (Gupta et al., 2011).

AI philosopical question - humans can do it, babies, dogs can do it - why can't a computer

Challenging problem: Human pose estimation inherits all the difficulties of object recognition - it shows off computation

For all these reasons, the task of estimating a human's pose from visual input is a worthwhile and timely problem.

Figure 1.1: An example illustrating the pose estimation problem, formalized in Problem 1.

1.1 Problem Statement

Here we formalize our problem definition input, output and computational requirements as follows:

Problem 1 (2D human upper-body pose estimation)**.**

Input: *A single RGB image or RGB video sequence containing the rough location and scale of a person in every frame, with no additional information.*

Output: *Line segments describing the major anatomical parts {left and right upper arms, left and right lower arms, torso, head} in pixel coordinates.*

Requirements: *Computation time and space polynomial in the number of input pixels and number of output parts.*

Importantly, we concern ourselves only with 2D (two dimensional) input. This makes the task much more challenging than when using additional sensors, such as in Microsoft's Kinect capture system (Shotton et al., 2011) where depth information and hence reliable knowledge of the background can be used. However, our limited-sensor problem also means it can be applied in more general settings: pose estimation outdoors and the wealth

of archival images and footages already stored on personal computers, libraries, and photo and video sharing web sites.

Furthermore, we do not assume any additional information, such as knowledge of the foreground, background, clothing, lighting, indoor or outdoor, etcetera. All these factors work to confound estimation by introducing appearance artifacts. We casually refer to out general setting as pose estimation *in the wild*, to stress the fact that the data sets we consider feature unconstrained foreground and background (or nearly unconstrained, when the input is from TV shows).

Also of note, we only consider the upper body, although all methods and models discussed in this work can be extended to full body processing (*i.e*. including hips and upper and lower legs). In fact, most of the models and tools develope in this work can be applied to other articulated objects, and in general, other domains in which estimating the instantiation of interacting parts (*e.g*., handwriting recognition, or gene sequencing). We focus on the upper body human pose in this work because (1) most interesting pose variation occurs in the upper body, (2) there is a vast amount data of people's upper bodies from TV shows, movies and images where lower halves are not visible, and (3) there is little extra knowledge to be learned about pose estimation by including the lower body parts, while increasing the computation time of all models at least linearly.

Finally, we restrict ourselves to polynomial running time. The space of all possible poses is exponential in the number of parts, and in some sense, the ideal thing to do if computation were free is enumerate all possibilities and score them holistically. However, this is simply not feasible, and we are forced to make conditional independence assumptions between certain parts to achieve tractability.

1.2 Intrinsic difficulties

Human pose estimation in the wild is an extremely challenging problem. It shares all of the difficulties of object detection, such as confounding background clutter, lighting,

Figure 1.2: Some of the perceptual challenges in human pose estimation. Large variations in lighting, pose, viewpoint, foreshortening, relative scale and clutter all work to confound pose estimation. See §1.2.1

viewpoint, and scale, in addition to significant difficulties unique to human poses. We are forced to reason over an enormous number of plausible poses for each image, making this a very compuationally demanding problem. In this section we go over the intrinsic difficulties of this problem, both from a vision and computational standpoint.

1.2.1 Perceptual issues

This is a difficult problem primarily because the appearance of pose is largely unconstrained, making it highly variable with multiple appearance modes. Some issues are as follows, illustrated by Figure 1.2.

Lighting: Images of pose can be taken indoor or outdoor, making not only the mean intensity of the image variable (signal bias), but also contrast (signal gain). This issue is well studied in computer vision, and to an extent, features have been developed to be invariant to lighting, *e.g.*, HoG (Dalal and Triggs, 2005b), but require harsh quantization and local normalization of edge energy information.

Viewpoint and pose: Humans can look very different depending on the angle they are to the camera. The global "twist" (rotation about the length-of-body axis) which determines the degree of frontal versus profile stance of the person, can be somewhat mitigated by coarse person detectors (Andriluka et al., 2010). However, the body can also go through radical appearance changes due to the articulation of the limbs, forcing practical systems to decompose the modeling into the most basic, articulation invariant components as basic units: limbs and joints.

Relative scale: Although we assume as input a detected person at a rough global scale, we have still have a large variation in the scale of parts in two different ways: In any particular person, the ratio of limb lengths may not be consistent; *e.g.* a baby's proportions versus an adults. Across people, there is also a large differences in the geometries of parts, based on gender, body type (fat, skinny, muscular), and age. These further contribute to the variability in appearance.

2D projection: The fact that we are working with images that are projections of the real world lead to further difficulties. Foreshortening makes estimating the length of the limb in 2D coordinates even more difficult, and changes the appearance. Self-occlusions and foreground occlusions make a part invisible and are very hard to determine without further scene or depth knowledge. Finally, it is inherently ambiguous to map from 2D pose to 3D real world coordinates (even up to an unknown global scale factor), discussed further

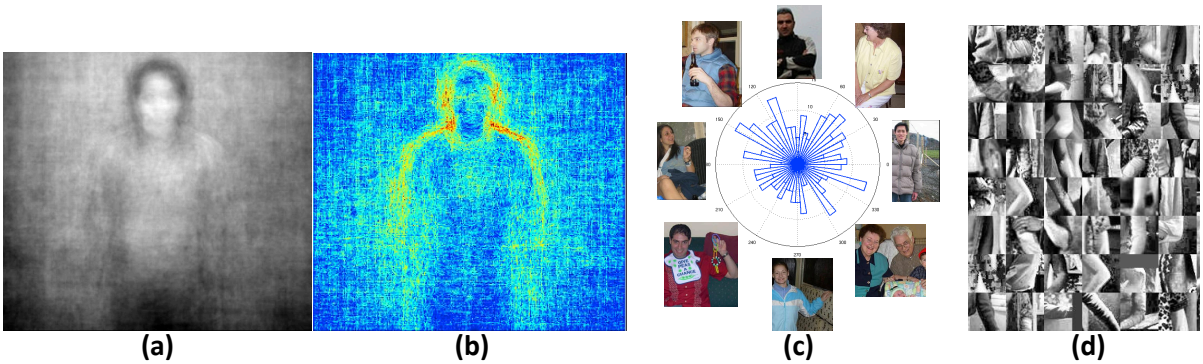

Figure 1.3: Some illustrations of variation in appearance in the PASCAL Stickmen dataset. (a) An average of the dataset in grayscale. (b) Average of sobel edges over dataset. (c) Distribution of inner angle made between upper and lower arm, with examples for $0°, 45°, 90°, 135°, 180°, 225°, 270°$ and $315°$. (d) A random sampling of 100 left elbows from the Buffy Stickmen pose dataset, removing color and intensity bias, to illustrate the huge variety of appearance due to clutter, motion blur, clothing, body type, and pose.

in §3.2. This makes it difficult to model priors on arm length, as we are forced to measure and reason about lengths on the 2D pixel grid.

Clothing: There is potential for even more variation in clothed people than there is in the space of naked bodies. Not only is clothing responsible for foreground clutter, it also can be considered an occluder which hides parts (*e.g.* baggy clothing, skirts, ponchos) and can break assumptions about left-right appearance symmetry (*e.g.* an asymmetric shirt).

Background clutter: **TODO:** forgot this one

1.2.2 Computational issues

To operationalize Problem 1, let x be the input image pixels, and y be a representation of the output predicted pose. Then a general solution to Problem 1 would take the form of a *scoring function* $h(x, y)$ which evaluates the quality of any estimated pose y in the

image x. We can define the "best" pose as the highest scoring: $y^\star = \arg\max_y h(x, y)$ (or $y^\star = \arg\sup_y h(x, y)$ if y is infinite dimensional, *e.g.* continuous). Then Problem 1 is satisfied if this determination of the maximizer can be done in polynomial time. There are two sources of intrinsic computational complexity within this framework.

Complexity of the input: For the reasons outlined in §1.2.1, there are an astronomical number of different inputs x that can map to the same true pose y, due to changes in the background, foreground, scene properties, camera effects, and variations in human pose, bodies and clothing. This makes modeling the space of poses from the input extremely difficult. The problem is inherently multi-modal, in the sense that radically different appearances are equally valid input representations of any particular pose. For a few different illustrations of the variability of a dataset, see Figure 1.3.

To deal with this complex problem, we are forced to either design features that are invariant to the multi-modal nature of the problem (*e.g.*, a generic patch-based arm detector based on coarse edges, or geometric features based on relative part coordinate systems), or to partition the space and model multiple modes separately. In the case of the latter approach, we are faced with other difficult decisions regarding model complexity: how to define modes and a notion of locality, and what the right trade-off is between the richness of the model and the error in fitting the model at different modes with a finite amount of training data.

Complexity of the output: The enormous combinatorial space of possible output poses is a second source of computational complexity. A typical discretization of the state space of human poses is an 80×80 spatial grid of part locations at 24 possible angles (Felzenszwalb and Huttenlocher, 2005), resulting in an output space that is roughly 150000 possibilities for each part, and thus $150000^6 \approx 10^{30}$ for the joint output space of all 6 parts [1].

Enumerating all possibilities for a joint configuration of all parts is clearly not feasible. At the other extreme, we could ignore part interactions, and estimate the pose of each part separately—a task which instead has 6×150000 possibilities, which is very

[1] In the case of continuous spaces, the output space is infinite (infinitely precise), and to maintain tractability the form of $h(x, y)$ is typically analytical with a closed-form maximum, mean and/or mode.

computationally cheap. However, individual part detection is extremely difficult for body parts (Andriluka et al., 2009) due to the wide range of appearance and lack of discriminating features.

Between the two extremes of (1) estimating parts in isolation and (2) enumerating all possible joint pose configurations, there lies a family of models $h(x, y)$ that consider *some* part interactions, but not all. The simplest of these is a *first-order* model which looks at pairs of part interactions at a time, and the graph of part interactions forms a tree structure. This compromise between a full model of every part and a decoupled model of independent parts will be the basic model building block throughout this work.

In such a first-order, or *pairwise* model, the basic bottleneck operation is to evaluate the quality of a pair of parts at a time. The model combines all such pairwise scores together to determine the optimal global pose. This scoring requires $150000^2 \approx 1$ billion possibilities to consider, which is large but just small enough for modern machines to handle with some additional model restrictions, detailed in §3.

TODO: reduce from NP-hard problem?

1.3 Contributions of this thesis

Due to the computational issues discussed above, previous work in pose estimation has resorted to a model of pose that considers, at most, pairwise interactions between parts, in a specially restricted form:

$$h(x, y) = \sum_{i \in \mathcal{V}_\Upsilon} \phi_i(x, y_i) + \sum_{i,j \in \mathcal{E}_\Upsilon} \phi_{ij}(y_i - y_j)$$

where the network of part pairwise interactions is described by a tree structured graph $\Upsilon = (\mathcal{V}_\Upsilon, \mathcal{E}_\Upsilon)$. The first terms $\phi_i(x, y_i)$ how likely a part is to be placed at location y_i, independent of other parts. The pairwise term $\phi_{ij}(y_i - y_j)$ simply measures the geometric compatibility between pairs of parts. Importantly, this pairwise term is *blind to the image content*. As we will see in §3, this simple form has been historically necessary in order to find the highest scoring global pose quickly.

Such a simplistic pairwise would be admissible if the individual part terms were sufficiently accurate on their own. However, for reasons discussed in §1.2.1, individual part detector scores are extremely weak: they must work in isolation and generalize to limbs in all settings of backgrounds, foregrounds, articulation, and environment.

As a result, the above model is effectively trying to piece together parts of a person, when the parts themselves are extremely ambiguous. As an extreme analogy of this, it is like attempting to put together a jigsaw puzzle in which the pieces themselves are very generic, and to determine whether two pieces fit together, you can only observe their boundary geometry, and *not* the image content on their faces.

1.3.1 Image-dependent interactions in a tree structured model, §4

Instead, we wish to actually *exploit* image content when modeling pairwise interactions. In the puzzle analogy, this would allow us to fit pieces together based on their color similarity and continuous contours across the connection boundary. We exploit these same cues for determining whether limbs go together in an image, as well as additional cues such as region support and multi-modal descriptions of geometry.

To this effect, we propose a more general model of human pose, of the form

$$\boxed{h(x,y) = \sum_{i \in \mathcal{V}_\Upsilon} \phi_i(x, y_i) + \sum_{i,j \in \mathcal{E}_\Upsilon} \phi_{ij}(x, y_i, y_j)} \tag{1.1}$$

This turns out to be computational infeasible using standard tools and techniques. In light of this, we propose a *cascade* of models to focus computation on portions of the state space that are more promising. The key insight is that there are many pose possibilities that are easy to reject as incorrect with a simple model, and we are then able to freely apply a richer model on the possibilities that remain. This is illustrated in Figure 4.1.

We develop and analyze *structured prediction cascades*, and apply them to the problem of 2D pose estimation, allowing us to use a model in the form of Equation 1.1 at every level of the cascade. Importantly, we provide a novel learning objective for the cascade of

models so that parameters of the model are tuned to specifically to filter out a significant proportion of states at any intermediate level.

1.3.2 Image-dependent interactions in a general graph, §5

The model we propose in the previous section works for any tree-structured model. However, in any tree model, we fail to capture important interactions between parts, both *within* a single frame (*e.g.*, color constancy between left/right symmetric parts to model clothing) and *across* frames when dealing with tracking multiple parts and their interactions over time in video. We address this with a generalization of Equation 1.1:

$$\boxed{h(x,y) = \sum_{i \in \mathcal{V}_G} \phi_i(x, y_i) + \sum_{i,j \in \mathcal{E}_G} \phi_{ij}(x, y_i, y_j)} \tag{1.2}$$

where $G = (\mathcal{V}_G, \mathcal{E}_G)$ is a general graph, not just a tree. Determining the best possible answer $\arg\max_y h(x,y)$ over a general graph G is known to be NP-hard (Koller and Friedman, 2009)—exponential in the number of frames of video. We provide an approximate approach which decomposes Equation 1.2 into a collection of subtree graphs, whose union of edges is equal to $\mathcal{E}_G$. This allows us to exploit all the interesting interaction terms in the original model with efficient inference in each subtree, thanks to the structure and the use of our cascade approach developed in §1.3.1. This allows us to exploit all the cues we used in the tree-based model, and in addition, cues based on color symmetry across the body, and temporal appearance and location persistence information. We propose and investigate empircally different methods of reaching a consensus between the subtrees.

We evaluate our approach on a new video dataset, the first of it's kind in tracking human pose in the wild without any assumed extra knowledge. We show our proposed model and approximation scheme is beneficial, beating the state-of-the-art in pose estimation systems.

1.3.3 Multi-modal interactions, §6

The aforementioned models focus attention on increasing the quality of features and increasing the number of modeled part interactions. The hope is that these more expressive models do a better job at capturing the inherently multi-modal appearance space of poses, by better separating the true pose configurations from false alarms. This somewhat addresses the issue of non-linearity in lower-dimensional feature spaces, *e.g.*, using only edge information.

Complementary to the models described by Equation 1.1 and Equation 1.2, we propose to capture this nonlinearity directly as follows:

$$\boxed{h(x, y, z) = \sum_{i \in \mathcal{V}_G} \phi_i(x, y_i, z) + \sum_{i,j \in \mathcal{E}_G} \phi_{ij}(x, y_i, y_j, z)} \tag{1.3}$$

where now the goal is to determine not only the best pose layout y, but also which *mode* z the pose belongs to: $y^\star, z^\star = \arg\max_{y,z} h(x, y, z)$. Each mode need only model a portion of the pose space, which we define by radii in appearance and pose centered around training examples. This means we get up to thousands of exemplars to form a basis for a discriminative pose space. Instead of fitting the parameters of one monolithic model to cover all possible modes, here we learn separate parameters for each mode, allowing us to learn more precise descriptions of appearance and geometry.

1.3.4 Summary of contributions

In summary, the work detailed in this thesis contributes the following to the fields of machine learning and computer vision, specifically their intersection for human pose estimation:

- New models of human pose that model interactions between parts as a function of image data.
- Computational innovations that enable learning and inference in these models, which are naïvely intractable: structured cascades and tree ensemble methods.

- A variety of new features and feature types not typically applied to pose estimation. Some of these are bottom-up type features complementary to the traditional edge-based cues.
- State-of-the-art results on the public Buffy and Pascal Stickmen single frame datasets, and our introduced Videopose video sequence dataset.

1.3.5 Published work supporting this thesis

Part I

Preliminaries

Chapter 2

Structured Prediction

In this section we lay out the basic definitions and tools necessary for data-driven modeling of *classification problems*. In the most general setting, the goal is to learn a mapping, or classifier h from a hypothesis class $\mathcal{H}$ from a set of input variables x to a set of discrete output variables y.

We concern ourselves with a specific setting: x lies in a (possibly high-dimensional) vector space (most commonly in this work, the input image pixels) and the target lies in a discrete n-dimensional space:

$$y \in \{0, 1, \ldots, k\}^n.$$

We will refer to the set $\{0, 1, \ldots, k\}$ as the set of labels, *label set*, or *state space* for a dimension of y. Each dimension of y will be referred to as y_i. For example, in a simple image classification task to determine whether an image has a dog in it, x could be an image's pixels, and $y \in \{dog, not\ dog\} \cong \{0, 1\}$, an instance of *binary classification*. In this case, $k = 2$ and $n = 1$. In *multiclass classification*, $k > 2$ and $n = 1$, *e.g.*, predicting the handwritten digits $0, \ldots, 9$ or the weather $\{sunny, cloudy, rainy\} \cong \{0, 1, 2\}$. Finally, in *structured prediction*, the output y is a vector: $n > 1$.

We will discuss four major components necessary for learning and using machine learning classifiers in this chapter.

- An **inference procedure** to determine the most likely label for a fixed test example

symbol	definition
$x \in \mathcal{X} = \mathbb{R}^d$	input variables
$y \in \mathcal{Y} = \{0, 1, \ldots, k\}^n$	output variables
$h(x, y) : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{Y}$	classifier

Table 2.1: Machine learning notation.

x:

$$h(x) \triangleq \arg\max_{y \in \mathcal{Y}} h(x, y). \tag{2.1}$$

- The **hypothesis class** $\mathcal{H}$ from which to obtain our classifier h.
- A **learning loss function** $\mathcal{L}(\cdot)$ which assesses the quality of any particular classifier $h \in \mathcal{H}$.
- A **learning algorithm** to find the minimizer of $\mathcal{L}$:

$$h = \arg\min_{h' \in \mathcal{H}} \mathcal{L}(h'). \tag{2.2}$$

2.1 Generalized linear classifiers

There are many possible parametric and non-parametric choices of hypothesis classes $\mathcal{H}$ to consider. One of the easiest to represent, learn and analyze is the hypothesis class of *generalized linear models*:

$$h(x, y) = \sum_{i=1}^{d} w_i f_i(x, y) \ = \mathbf{w} \cdot \mathbf{f}(x, y) \tag{2.3}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a vector of linear parameters of our model and $\mathbf{f}(x, y) \in \mathbb{R}^d$ is a vector of features that depend on the input and output variables. Thus the model is simply a weighted sum of features to obtain a real valued score, and $\mathcal{H} = \mathbb{R}^d$.

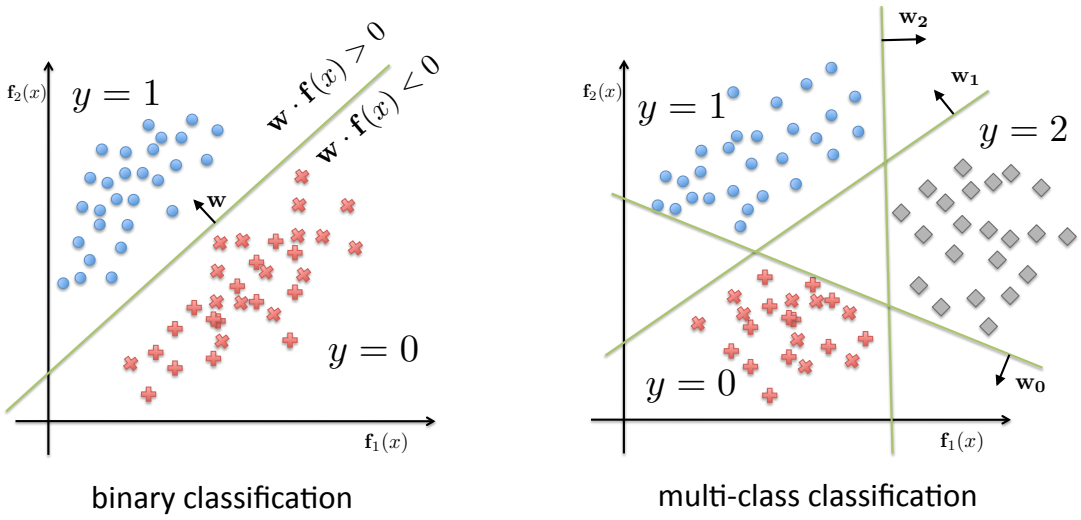

Figure 2.1: Binary and multi-class classification. On the left is shown a binary labeling problem in 2 dimensions, with a linear classifier which perfectly classifies the data. On the right is shown a 3 class classification problem, with a linear classifier for each class, each of which perfectly separates the correct label from the two incorrect labels.

2.1.1 Binary and multi-class classifiers

In binary ($y \in \{0, 1\}$) and multi-class ($y \in \{0, \ldots, k\}$) classification problems, there is a simple and intuitive geometric interpretation to linear classifiers. In two dimensions, the classifiers describe a line, in three dimensions, they describe a plane, and in general, they describe a *hyperplane* which partitions the feature space (for a particular label) into two halfspaces.

In the case of binary classification, observe that for binary classification, we need only look at which side of a hyperplane a point lies on in a particular feature space to determine

the label:

$$h(x) = \mathbf{1}\left(h(x,1) > h(x,0)\right) \tag{2.4}$$

$$= \mathbf{1}\left(\mathbf{w}\cdot\mathbf{f}(x,1) > \mathbf{w}\cdot\mathbf{f}(x,0)\right) \tag{2.5}$$

$$= \mathbf{w}\cdot\left(f(x,1) - f(x,0)\right) > 0 \tag{2.6}$$

$$= \mathbf{w}\cdot\tilde{f}(x) > 0 \tag{2.7}$$

The vector $\mathbf{w}$ describes a hyperplane via its surface normal. The test $\mathbf{w}\cdot\tilde{f}(x) > 0$ determines which side of the hyperplane an example lies on, which also corresponds to its predicted label, as in Figure 2.1, left. When $k > 2$, we can also transform feature spaces and interpret the linear classifier as k separating hyperplanes in d dimensions, one hyperplane to separate each label from the rest. Figure 2.1, right illustrates an example with $k = 3$ and $d = 2$.

2.2 Pairwise structured models

In structured prediction, $|\mathcal{Y}| = |\{0,\ldots,k\}^n|$ is exponential in n and typically enormous. Due to computational and modeling considerations to be discussed, we assume that the full model $\mathbf{w}\cdot\mathbf{f}(x,y)$ *decomposes* into factors, or *cliques* which involve overlapping sets of variables.

$$\mathbf{w}\cdot\mathbf{f}(x,y) = \sum_{c\in\mathcal{C}} \mathbf{w}_c\cdot\mathbf{f}_c(x,y_c) = \sum_{c\in\mathcal{C}} \phi_c(x,y_c), \tag{2.8}$$

where we use the shorthand for cliques $\phi_c(\cdot) \triangleq \mathbf{w}_c\cdot\mathbf{f}(\cdot)$. We can encode the sets of factors and their overlap relations in general using a *factor graph* (Koller and Friedman, 2009). The number of different settings and representations for structured models is vast and varied and outside the scope of this work.

For our purposes, we focus on structured problems with at most *pairwise structure*, where $|c| \leq 2 \;\; \forall c \in \mathcal{C}$. This simplification does not result in loss of generality, as factors involving 3 or more variables can always be converted into pairwise or unary factors with an expanded label set consisting of the Cartesian product of each variable's state space.

For example, a factor involving the variables y_1, y_2 and y_3 with state space $\{1, \ldots, k\}^3$ can be converted into a single variable $y_{123} \in \{1, \ldots, k^3\}$.

We can represent a pairwise structured model as a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ where each variable y_i corresponds to a vertex in the graph's vertex set $\mathcal{V}_G$, and each edge in $\mathcal{E}_G$ corresponds to two variables y_i, y_j being involved in a pairwise factor ϕ_{ij}. We can then decompose our classifier into the special form

$$\mathbf{w} \cdot \mathbf{f}(x, y) = \sum_{i \in \mathcal{V}_G} \mathbf{w}_i \cdot \mathbf{f}(x, y_i) + \sum_{i,j \in \mathcal{E}_G} \mathbf{w}_{ij} \cdot \mathbf{f}(x, y_{ij}) \tag{2.9}$$

$$= \sum_{i \in \mathcal{V}_G} \phi_i + \sum_{i,j \in \mathcal{E}_G} \phi_{ij}. \tag{2.10}$$

There is a huge amount of research dedicated to models of this form originally stemming from statistical mechanics, where it was first used to determine the spin of particles connected in a grid, and Equation 2.10 describes the log of the energy of the system. When given a probabilistic interpretation as we will see in §2.2.1, this type of model is known as a *pairwise Markov Random Field* (MRF). Because of such historical intuitions as a model for energy, we refer to the different terms as *potentials*, and Equation 2.10 as the negative of an *energy function* which we seek to minimize. Terms of the form $\phi_i = \mathbf{w}_i \cdot \mathbf{f}_i$ we will refer to as *unary potentials*; $\phi_{ij} = \mathbf{w}_{ij} \cdot \mathbf{f}_{ij}$ are *pairwise potentials*.

Pairwise MRF examples

Our primary application of interest in this work is human pose estimation, whose modeling as an MRF will be discussed in detail in §3. However, to first give motivation and intuition about how and why to model problems via a pairwise MRF, we present examples here.

Wandering robot For example, imagine we are tracking the location of a robot on a map with 100 locations over 100 time steps, see Figure 2.2-a. We have no idea what the robot's intent is or where he starts out in the world (the "kidnapped robot" scenario), only that he can move only to an adjacent grid position at each time step (*i.e.*, cannot teleport). We have noisy sensor readings $x = [x_1, \ldots, x_{100}]$ for every time step. The output y is a sequence of locations the robot visited in all 100 time steps, one out of $\mathcal{Y} = 100^{100} = 10^{1000}$

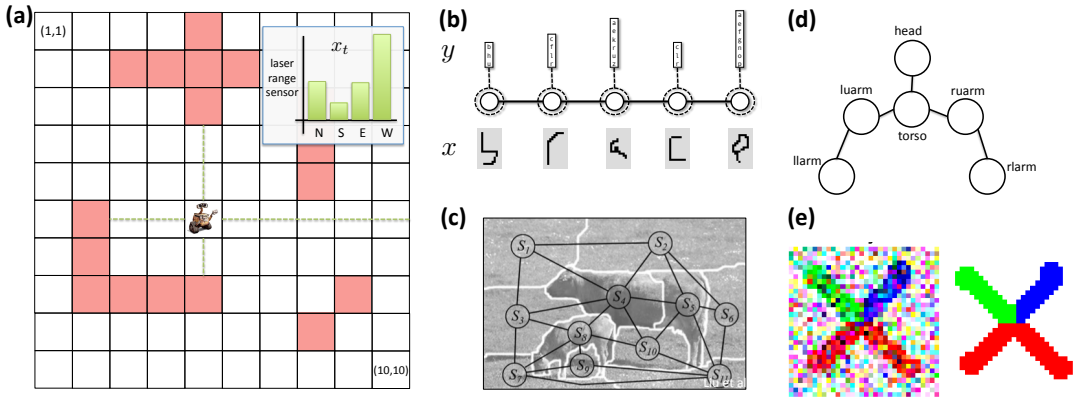

Figure 2.2: MRF examples. (a) A robot localization problem on a 10×10 grid. (b) Handwriting recognition. (c) Scene labeling. (d) Human pose estimation. (e) Image denoising. See text for details.

possibilities. By comparison, the estimated number of atoms in the universe is only 10^{80}. A naïve approach would be to model this as a multi-class problem with 10^{1000} labels, and try to directly learn a mapping $h(x, y) = \mathbf{w} \cdot \mathbf{f}(x, y)$ for all possible y. Importantly, there is a lot of *structure* to this problem, the most obvious being that many outputs y are just not valid, due to the fact that the robot can only move to an adjacent location at each step.

The key insight to be made is the following: if we know where the robot is at time t, this is tremendously useful for determining the robot's next location at time $t + 1$. In addition, knowing where the robot was at $t - 1$ also helps localize it at $t + 1$, but with diminishing returns—we guess the robot is in roughly the same spatial neighborhood, and also exploit knowledge of its velocity. Going further into the past, there is increasingly less information to help localize where the robot is at time $t + 1$ if we are told where it was many steps ago, say, $t - 10$. In general, we can make the simplifying assumption that *given the recent past, the future is independent of the distant past*. Using this intuition, we can model the problem as a *chain* MRF:

$$\mathbf{w} \cdot \mathbf{f}(x, y) = \sum_{t=1}^{100} \phi_t(x, y_t) + \sum_{t=1}^{99} \phi_{t,t+1}(x, y_t, y_{t+1}) \tag{2.11}$$

where the graph structure is a node for the output variable at each time step y_t, and we only consider pairwise factors over adjacent time steps $\phi_{t,t+1}$, creating a chain of variable interactions. The unary potentials ϕ_t can model the likelihood of being in each grid location at time t based on sensor readings, and the pairwise potentials can model how likely it is to transition from any location y_t to any location y_{t+1}. The pairwise term in such tracking problems is typically represented as a *transition matrix*. In this robot localization problem the transition matrix would be very sparse, since any location can only transition to adjacent grid locations.

In other structured problems, it also makes sense to make *independence assumptions* about which dimensions of y are assumed dependent on each other and their interactions should be directly modeled.

Handwriting recognition In this problem, the output at each position is which letter of the alphabet is written given handdrawn letter images x ($k = 26$ if only considering 'a',...,'z', or roughly 50 if considering all alphanumerics plus punctiation). In this problem, clearly adjacent letters in a document are highly correlated (*e.g.*, adjacent outputs such as 'ab' and 'he' are likely, but 'hb' is not), but depend very little on letters far away in the word, sentence or even document. Practitioners again typically model this as a chain, seen in Figure 2.2-b.

Scene labeling In scene labeling, the goal is to label coarse segments of an image with scene types ("sky","grass","building","cow", etcetera; k is on the order of tens) given the image as x. The typical assumption made is that adjacent regions output directly depend on each other, whereas the label of spatially distant segments have weak or no interactions and are not modeled (Cour et al., 2005). The MRF model thus connects adjacent segments in the image, giving us a cyclic planar graph Figure 2.2-c.

Human pose estimation Knowing the location of the left shoulder is strong cue for where the left elbow should be, since they are kinematically coupled in the real world, but only a weak indicator of where the right wrist should be. The typical representation is to describe the human layout as a tree graph corresponding to the kinematic skeleton, see Figure 2.2-d

and §3.

Image denoising Here the input x is an image with a small set of labels that are corrupted by noise, the goal is to determine the uncorrupted original image y the same size as x. Here k is typically 2 or a small set of indexed colors. Practitioners typically model this problem with a grid graph, where variable $y_{r,x}$ corresponding to pixel label at row r, column c is connected to (is in pairwise factors with) $y_{r+1,c},\ y_{r-1,c},\ y_{r,c+1},$ and $y_{r,c-1}$. See Figure 2.2-e.

You may object that some of the decomposition assumptions made in the above examples are somewhat extreme. However, they are typically seen as forgivable thanks to the greater simplicity of modeling only local, overlapping interactions. Even more enticing is the reduction in computation they allow, discussed in §2.3.

2.2.1 Probabilistic interpretation

We have motivated most of this chapter via first principles about linear classifiers. However, much of the development of models and methods originally came from the probabilistic model community. From another perspective, our model describes a *log-linear* probabilistic formulation of the posterior probability $p(y|x)$:

$$p(y|x) = \frac{\exp\left[\mathbf{w} \cdot \mathbf{f}(x, y)\right]}{\sum_{y \in \mathcal{Y}} \exp\left[\mathbf{w} \cdot f(x, y)\right]} = \frac{1}{Z(x)} \exp\left[\mathbf{w} \cdot \mathbf{f}(x, y)\right] \tag{2.12}$$

It can be shown (Jaynes, 1963) that this particular form of $p(y|x)$ is the distribution with maximum entropy, subject to the constraints that feature expectations with respect to $p(y|x)$ match empirical feature expectations. This principle is based on the desire to have our model make as few assumptions as possible (most entropic) about the observed data.

Because we are typically only interested in the most likely y—here, the *maximum a posteriori* or MAP assignment—it is sufficient to find the $\arg\max$ of a simplified quantity and not worry about normalization by $Z(x)$:

$$\arg\max_y p(y|x) \tag{2.13}$$

$$= \arg\max_y \log p(y|x) \tag{2.14}$$

$$= \arg\max_y \mathbf{w} \cdot \mathbf{f}(x, y) - \log Z(x) \tag{2.15}$$

$$= \arg\max \mathbf{w} \cdot \mathbf{f}(x, y) \tag{2.16}$$

A distribution $p(y|x)$ modeling a structured y that decomposes over factors as in §2.2 takes the form:

$$p(y|x) \propto \exp\left[\sum_{i \in \mathcal{V}_G} \phi_i + \sum_{i,j \in \mathcal{E}_G} \phi_{ij}\right] = \prod_{i \in \mathcal{V}_G} \exp \phi_i \prod_{i,j \in \mathcal{E}_G} \exp \phi_{ij} \tag{2.17}$$

Historically and in other settings, terms $\exp \phi_c$ were assumed to be themselves proper joint $p(y_c, x)$ or posterior $p(y_c|x)$ distributions over subsets of random variables c, but we assume no such restriction in our case.

2.3 Inference

The inference problem is to determine the highest scoring possibility out of all possible outputs in $\mathcal{Y}$:

$$y^\star = h(x) = \arg\max_{y \in \mathcal{Y}} h(x, y). \tag{2.18}$$

We use the convention h to mean *hypothesis* or *hypothesis class*, originating from the machine learning community. In the spirit of optimization, we also view this as a *scoring function* and use the synonomous notation s to denote the score:

$$s(x) = \max_{y \in \mathcal{Y}} s(x, y) \triangleq h(x, y) \tag{2.19}$$

When $\mathcal{Y}$ is low-dimensional, brute-force search is easy enough: evaluate $h(x, y)$ for all possible y, and return the highest scoring. This can be done for binary classification—where we need only check which side of a separating hyperplane an example lies on

(§2.1.1)—and multi-class classification, where we evaluate each of k labels do determine the highest scoring.

However, in the most general setting, $\mathcal{Y}$ is exponential in k: $|\mathcal{Y}| = k^n$, and we can't hope to enumerate all possible outputs in $\mathcal{Y}$ to find the highest-scoring. Thankfully, the decomposable structure we impose on our models, discussed in §2.2, allows us to find a solution time polynomial in k and linear in n, rather than exponential in n.

As a simple example of this, consider our wandering robot problem where we make the common assumption that we only model temporally adjacent pairs of output variables together, making a chain of variable dependencies:

$$\mathbf{w} \cdot \mathbf{f}(x, y) = \sum_{t=1}^{100} \phi_t(x, y_t) + \sum_{t=1}^{99} \phi_{t,t+1}(x, y_t, y_{t+1})$$

Observe now how computing the maximum score of the classifier *also decomposes* for this simple problem:

$$\max_{y \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{f}(x, y) = \max_{y_1, \ldots, y_{100}} \sum_{t=1}^{100} \phi_t(x, y_t) + \sum_{t=1}^{99} \phi_{t,t+1}(x, y_t, y_{t+1}) \tag{2.20}$$

$$= \max_{y_{100}} \left[\phi_{99,100} + \phi_{100} + \ldots \max_{y_3} \left[\phi_{3,4} + \phi_3 + \max_{y_2} \left[\phi_{2,3} + \phi_2 + \max_{y_1} \phi_{1,2} + \phi_1 \right] \right] \ldots \right] \tag{2.21}$$

In Equation 2.21, we see that the max over each dimension of y can be nested and needs only reason over at largest a pairwise factor at a time. In our robot problem, each pairwise factor is only 100^2 numbers to consider and each unary factor is of size 100. Thus the total amount of numbers that need to be examined in this problem is roughly 10^6, an astronomical improvement over the complete space of 10^{1000}. In general the bottleneck of inference is the max over the pairwise factors of size k^2, and thus inference is $O(n \cdot k^2)$.

In general, when an MRF model forms a tree, we can always exploit the structure for exact efficient inference (Koller and Friedman, 2009). Using the same trick we used in the specific example of the robot localization problem Equation 2.21, we can perform inference by nesting the $\max$ operator over pairs of variables at a time, as well as storing the $\arg\max$ maximizer of each $\max$ operation. For completeness, the algorithm is in

Algorithm 1, commonly referred to as max-sum *message passing, belief propagation*, or *viterbi decoding*. The algorithm results in inference that is $O(nk^2)$ computation rather than k^n.

Algorithm 1 Max-sum message passing to solve

$$\arg\max_{y \in \mathcal{Y}} h(x, y) = \arg\max \sum_{i \in \mathcal{V}} \phi_i + \sum_{ij \in \mathcal{E}} \phi_{ij}$$

Input:

Factors $\{\phi_i\}, \{\phi_{ij}\}$

Tree graph G with (arbitrary) root node index r and topological ordering π, where $\pi_n = r$.

Ouput: $y^\star = \arg\max_y h(x, y)$

for $i = \pi_1, \pi_2, \ldots, \pi_n$ **do**

$m_i = \phi_i + \sum_{j \in \text{kids}(i)} m_{j \to i}$

if $i == r$ **then**

break

end if

$p = \text{parent}(i)$

$m_{i \to p} = \max_{y_i} \phi_{ip} + m_i$

$a_i = \arg\max_{y_i} \phi_{ip} + m_i$

end for

$y_r^\star = \arg\max_{[1 \ldots k]} m_r$

for $i = \pi_{n-1}, \pi_{n-2}, \ldots, 1$ **do**

$y_i^\star = a_i \left[y^\star_{\text{parent}(i)} \right]$

end for

2.4 Max-marginals

Recall the score of a joint part state y in input x is $s(x, y)$. Often we are interested in just the best scoring complete assignment $y^\star = \arg\max_y s(x, y)$ and its corresponding score $s^\star(x) = \max_y s(x, y)$, both obtainable from Algorithm 1. However, another extremely useful quantity which we rely on often is the notion of a *marginal* score. When treating our model as a log-linear conditional distribution $p(y|x)$ (see §2.2.1), we can consider the marginal distribution of a particular variable

$$p(y_i|x) = \sum_{y_j \text{ s.t. } j \neq i} p(y|x) \tag{2.22}$$

as a measure of the model's belief over the value of variable i.

Similarly, we explore the notion of a *max-marginal*, an analogous quantity for the $\max$ operator:

$$s_x^\star(y_i) \triangleq \max_{y'} s(x, y'), \text{ subject to: } y'_i = y_i \tag{2.23}$$

In words, the max-marginal score for y_i is the score of the best full assignment restricted to fixing variable i to be y_i.

A naïve way to compute $s_x^\star(y_i)$ for all states in our model is as follows: for each i and each y_i (in total nk possibilities), set $\phi_{ij}(x, y'_i, y_j) \to -\infty$ for all $y'_i \neq y_i$, and all j attached to i, then run Algorithm 1. This ensures that y_i is the "chosen" state for variable i, and we get its max-marginal score $s_x^\star(y_i)$. This strategy would cost in total $O(nk \cdot nk^2) = O(n^2k^3)$, which is not very satisfying.

It turns out we can compute max-marginal quantities for all nk variable-state possibilities in $O(nk^2)$ time using a forward *and* backward pass of message passing and careful bookkeeping, similar to Algorithm 1. In addition, we can also collect *witness statistics*, which keep track of the number of times different states have been used in any max-marginal satisfying assignment (and thus are a "witness" to the assignment). The *witness* for $s_x^\star(y_i)$ is

$$y^\star(y_i) \triangleq \arg\max_{y'} s(x, y'), \text{ subject to: } y'_i = y_i, \tag{2.24}$$

which is the maximizer which corresponds to maximum $s^\star_x(y_i)$. For completeness, the algorithm to compute max-marginal values and witness statistics is included in §?? in Algorithm ??.

2.5 Supervised learning

§?? In the supervised learning setting, we have access to a training set $S = \{(x^{(j)}, y^{(j)})\}_{j=1}^m$ of examples assumed to be sampled independently, identically distributed (i.i.d.) from some true joint distribution $(x, y) \sim P(X, Y)$. The standard supervised learning task is to learn a hypothesis $h : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{Y}$ that minimizes the average misclassification error (0/1 error) on the training set:

$$\mathcal{L}_{0/1}(h, S) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left(h(x^{(j)}) \neq y^{(j)}\right) \tag{2.25}$$

The linear hypothesis class we consider is of the form $h(x) = \arg\max_y h(x, y)$, where the scoring function $h(x, y) \triangleq \mathbf{w} \cdot \mathbf{f}(x, y)$ is the inner product of a vector of parameters $\mathbf{w}$ and a feature function $\mathbf{f} : X \times Y \mapsto \mathbb{R}^d$ mapping (x, y) pairs to d real-valued features, as discussed in §2.1.

Using a linear representation, we have the following optimization problem to learn our model from training data:

$$\underset{\mathbf{w}}{\textbf{minimize}}\ \mathcal{L}_{0/1}(h, S) = \tag{2.26}$$

$$\underset{\mathbf{w}}{\textbf{minimize}}\ \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left(\arg\max_y [\mathbf{w} \cdot \mathbf{f}(x^{(j)}, y)] \neq y^{(j)}\right) \tag{2.27}$$

The above optimization is extremely difficult to solve in high dimensions, because it is non-convex and discontinuous, leading to many poor local minima. In light of this, we introduce a *convex surrogate* to the indicator loss function $\ell(\cdot)$ to replace $\mathbf{1}(\cdot)$ with an

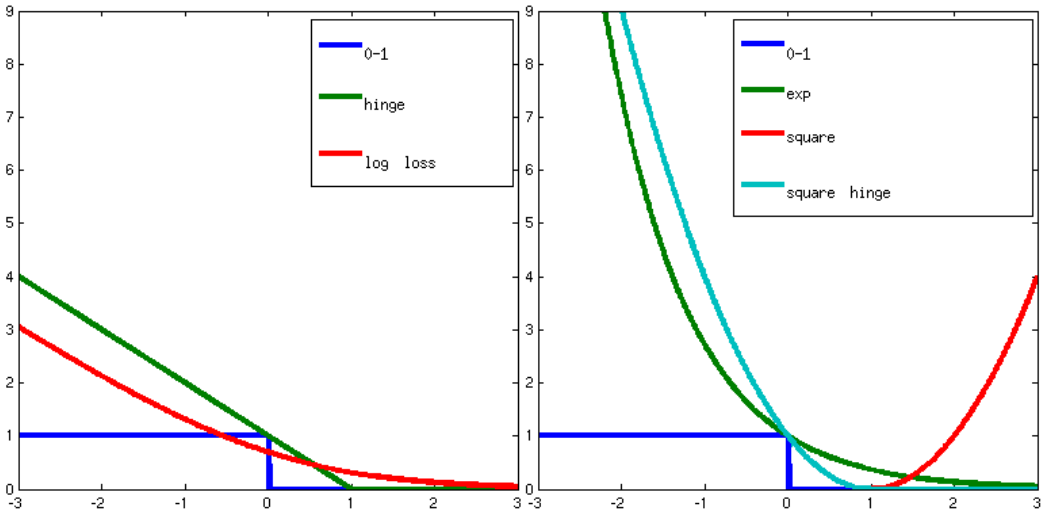

Figure 2.3: Convex surrogate supervised loss functions.

upper-bound (Bishop, 2006):

$$\underset{\mathbf{w}}{\textbf{minimize}} \ \mathcal{L}_\ell(h, S) = \tag{2.28}$$

$$\underset{\mathbf{w}}{\textbf{minimize}} \ \frac{1}{m} \sum_{j=1}^{m} \ell \left(\mathbf{w} \cdot \mathbf{f}(x^{(j)}, y), y^{(j)} \right) \tag{2.29}$$

There are many common choices for $\ell(\cdot)$, shown in Figure 2.3, leading to many different structured and non-structured machine learning algorithms. We choose to use the *hinge loss* for most of this work, which leads to a simple and intuitive stochastic optimization algorithm. The hinge loss is of the form $\ell(u) = \max(0, 1 - u) \triangleq [1 - u]_+$, giving us the following optimization problem:

$$\underset{\mathbf{w}}{\textbf{minimize}} \ \frac{1}{m} \sum_{j=1}^{m} \left[\arg\max_{y} \mathbf{w} \cdot \mathbf{f}(x^{(j)}, y) - \mathbf{w} \cdot \mathbf{f}(x, y^{(j)}) + 1 \right]_+ \tag{2.30}$$

Chapter 3

Pictorial structures: Pose estimation meets structured prediction

Classical pictorial structure models (PS models) are a class of structured models specifically proposed to represent 2D objects with parts which can articulate in a kinematically plausible way. The model was first proposed by Fischler and Elschlager (1973), and has been hugely popular in computer vision since computational innovations by Felzenszwalb and Huttenlocher (2005).

Pictorial structures take the form of a pairwise structured model as described in §**??**:

$$h(x, y) = \sum_{i \in \mathcal{V}_G} \phi_i(x, y_i) + \sum_{ij \in \mathcal{E}_G} \phi_{ij}(y_i, y_j) \tag{3.1}$$

with the important exception that *the unary terms $\phi_{ij}(y_i, y_j)$ are image-independent*—they are not a function of x.

In the PS model, the variables correspond to the major body parts. For upper-body pose estimation, y typically corresponds to [head, torso, left-upper-arm, left-lower-arm, right-upper-arm, right-lower-arm], and $y_i \in [1, \ldots, 80 \times 80 \times 24]$ represents a typical 80×80 discretized grid of spatial locations, and 24 discretized angles for each part. Thus one way to think about the representation for each part is a unit vector describing position and orientation of each limb.

Edge structure: In order to support tractable inference, the part interaction structure G must be a tree. The most common tree is formed by considering kinematic connections only: the lower-arm is connected to the upper-arm, the upper-arm to the torso, etcetera.

Unary potentials: The unary potentials $\phi_i(x, y_i)$ encode how likely a limb is at (location,orientation) y_i in the image. This can be thought of as a part detector applied to an image patch located at y_i. In practice these detectors are patch-based sliding window object detectors, applied at every location and orientation of the image. The typical representation is based on edge information in order to be invariant to lighting and color. Edges orientation and magnitude are often locally quantized and histogrammed to be more numerically and spatially stable, and invariant to local signal gain. Feature representations are discussed in detail in §7

Pairwise potentials: Pictorial structures assumes a restricted form of pairwise potential the depends only on the deformation between kinematially attached parts. In general, this is a quadratic stretching cost of the form

$$\phi_{ij}(x, y_i, y_j) = -||T_{ij}y_i - y_j - \delta_{ij}||_2^2 \tag{3.2}$$

where T_{ij} are rigid transformations (rotation, translation and scale) to place neighboring parts in a local coordinate frame, and δ_{ij} is the expected displacement between the parts. For example, $\phi_{ij}(x, y_{\text{luarm,llarm}})$ measures the squared Euclidean distance between the elbow locations according to the left upper arm and left lower arm variables.

Spring model interpretation The reasons for restricting the pairwise potential to be only a function of spatial deformation are primarily computational, as discussed in §**??**. In addition, this type of model is simple and intuitive to interpret as a "spring model" of object layout: the PS model can be interpreted as a set of springs at rest in default positions δ_{ij} stretched by displacement $T_{ij}y_i - y_j - \delta_{ij}$. The spring tightness is encoded by warping transformations T_{ij} which can scale the dimensions of the spatial Euclidean coordinate space. The unary terms pull the spring ends towards locations y_i with higher scores ϕ_i which are more likely to be a location for part i. Thus the spring model seeks a balance between confidence in individual part detectors, and the amount of deformation from a

default, 2D geometric prior of the human layout.

3.1 Sub-quadratic inference

As discussed in §**??**, the computation required to obtain the highest-scoring pose out of the exponentially many possibilities according to Equation 3.1 is $O(kn^2)$, using Algorithm 1. In pose estimation, the reason is intuitive: for each possible location of, say a left upper arm, we need to exhaustively search to find the best location of a corresponding lower arm, checking all such possibilities of left lower arms. Doing this for all left upper arms yields an exact n^2 search.

In practice, there is no need to check *all* possibilities of lower arms for each upper arm location; it is sufficient to check only in a reasonable spatial window around the mean displacement location δ_{ij}, because it is impossible for kinematically-connected object parts to be stretched too far. Even so, we must search over a fixed fraction of neighbor states for each part, which still scales quadratically with the resolution of the state space. In practice, for all $80 \times 80 \times 24$ locations of an object part, a state space window of $80/5 \times 80/5 \times 24$ possibilities need to be evaluated, yielding on the order of $(80 \times 80 \times 24)^2/25 = 943,718,400$ computations—nearly a billion—for a pair of parts.

The search operation described here is exactly that performed in the following lines of Algorithm 1, where i indexes a part, and j its parent in a topological ordering of the tree graph[1]:

$$m_{i \to j} = \max_{y_i} \phi_{ij} + m_i \tag{3.3}$$

$$a_i = \arg\max_{y_i} \phi_{ij} + m_i \tag{3.4}$$

The quantity $a_i(y_j)$ is the best placement of part i when placing part j at location y_j, using all the information from predecessor parts in the topological ordering. The quantity $m_{i \to p}(y_j)$ is the corresponding score for that placement.

[1] We substituted variable index p with j here in keeping with the notation in this section.

It turns out, as introduced by Felzenszwalb and Huttenlocher (2005), that when the pairwise term ϕ_{ij} takes the form it does for classical PS, as in Equation 3.2, then we can apply a *generalized distance transform* procedure to compute a_i and $m_{i \to j}$ over all possibilities for y_i and y_j in time linear in the number of states (versus the naïve quadratic time).

The generalized distance transform solves the problem

$$\mathcal{D}_q(p; f) = \min_q ||p - q||_2^2 + f(q) \tag{3.5}$$

for any discrete function $f(\cdot)$ where p and q are discrete scalar quantities with bounded domain. The solution to the above can be found via dynamic programming to keep track of the lower envelope of parabolas of the form $(p-q)^2 + f(q)$, operating over a dimension at a time (Felzenszwalb and Huttenlocher, 2006). By manipulating Equation 3.3, we can massage it into the form of Equation 3.5.

$$\max_{y_i} \phi_{ij} + m_i(y_i) \tag{3.6}$$

$$= \max_{y_i} -||T_{ij} y_i - y_j - \delta_{ij}||_2^2 + m_i(y_i) \tag{3.7}$$

$$= -\min_{y_i} ||T_{ij} y_i - y_j - \delta_{ij}||_2^2 + (-m_i(y_i)) \tag{3.8}$$

$$= -\min_{y_i} ||\tilde{y}_i - y_j||_2^2 + (-m_i(y_i)) \tag{3.9}$$

$$= -\mathcal{D}_{\tilde{y}_i}(y_j; -m_i) \tag{3.10}$$

Where $\tilde{y}_i$ is a transformed version of the 2D state space for y_i via the rigid transform of T_{ij} and a translation by δ_{ij}. In fact, the tricks used to compute the generalized distance transform efficiently work for any unimodal functions of geometric displacement. In particular, an L_1-norm transform can also be used.

TODO: relwork: julian's thing, branch and bound, hmm max thing

3.2 Limitations

There are some severe limitations to the classical PS model discussed in this section which motivate the contributions of this thesis.

3.2.1 2D representation for a 3D object

Inferring 3D pose from 2D input is inherently ambiguous. Even when the real world geometric specifications of the object are known (*e.g.*, the true lengths of limbs in centimeters, or their length ratios), there is ambiguity due to camera projectection—does the imaged limb appear shorter because it is pointing towards or away from the camera? In a solution provided by Taylor (2000), knowing the locations of n limbs' image coordinates and real world proportions results in a family of 2^n possible real-world solutions, up to an unknown scale factor. User-guided interfaces aided with this technology still take minutes of trial and error adjustments to get a plausible 3D pose from 2D (Bourdev and Malik, 2009b).

Thus it is a intrinsic problem that we must reason in 2D with 2D input (although when estimating pose in video, temporal consistency may help resolve some of the ambiguities). As a result, the 2D geometric priors are inherently weak. Fixing global scale of the person, it is only safe to say that a limb length ranges from some maximum distance (when parallel to the camera plane) down to 0 pixels (when the pointing straight at the camera). This also makes background and self-occlusion extremely difficult to model.

3.2.2 Cardboard people

Representing pose as limbs parametrized by location and orientation has been called a "cardboard people" representation, since at best limbs can be described by rectangles of fixed dimensions (from idealized cylinder representations of limbs projected into the image) (Ju et al., 1996a). Such a mode leaves no room for modeling foreshortening, and is forced to discretize the set of orientations to some coarse 10° to 15° increments. In fact, the system originally proposed by Felzenszwalb and Huttenlocher (2005) also considered

10 discretized values of scale for each part to handle foreshortening, but no recent publicly available system (Andriluka et al., 2009; Eichner and Ferrari, 2009; Sapp et al., 2010a,b, 2011) includes a scale for each part. The reasons are two-fold.

First, it is computationally more expensive. Ten scales per part results in a state space for each part that is 10 times larger. When using distance transform inference tricks as in §3.1, this makes inferences 10 times slower. If using standard max-sum inference (§**??**), it would be 100 times slower.

Second, there is very little signal in real world scenarios from which to infer foreshortening. In Felzenszwalb and Huttenlocher (2005) the system operated on foreground sillhouettes, making background clutter a non-issue. When severely distorted, it is extremely difficult to determine, even to the human eye, what is a foreshortened arm and what is simply background clutter. **TODO:** figure . This problem can be ameliorated when parsing human pose in video, in which we can track joints as the arm undergoes a foreshortening over a sequence of frames. We address these issues in §5.

3.2.3 Unimodal potentials

In order to achieve fast inference, modeling sacrifices have been made. The linear time max-sum inference for PS described in §3.1 via distance transforms is supported only if the pairwise potentials are a unimodal function of geometric displacement. This is quite a strong assumption, especially for representing the angle between, *e.g.*, upper and lower arms. Figure blah blha blah **TODO:** figure .

Furthermore, even the unary potentials are typically unimodal, in the sense that they are modeled as individual linear classifiers on an edge representation, of the form $\mathbf{w}_i \cdot \mathbf{f}_i(x, y)$. As a consequence, all the variations of part appearance due to pose, deformation, body type, lighting, foreshortening and clothing are being fit with a single linear model, resulting in a harsh quantization of the rich space of poses. The reasons for this are not only computational (the weights can be re-interpreted as a 2D linear filter, and $O(n \log n)$ convolution can be used to evalute them), but also statistical: using a more complex model

requires more training data for accurate estimation. This issue is the motivation for the model proposed in §6.

3.2.4 Image-independent interactions

Maybe the most unsatisfying property of the pictorial structure model is its effective blindness to image content when determining how to "piece patches together" in a geometrically plausible way to obtain the best scoring pose, due to the image-independent pairwise terms $\phi_{ij}(y_i, y_j)$. This is a necessity in order to achieve sub-quadratic inference, as discussed in §3.1. This means we are unable to encode in the pairwise term the likelihood that a pair of part hypotheses go together because they *look like* they go together; only that they go together because they fit together geometrically. Thus we are unable to express color, shape or region compatibility for a pair of parts in our model. Overcoming this limitation is a major contribution of this thesis, addressed in §4 and §5.

Part II

Models and methods

Chapter 4

Cascaded Pictorial Structures

4.1 Introduction

Pictorial structure models, first proposed by Fischler and Elschlager (1973) and outline in §3, are a popular method for human body pose estimation Andriluka et al. (2009); Felzenszwalb and Huttenlocher (2005); Fergus et al. (2005); Ferrari et al. (2008); Ramanan and Sminchisescu (2006). The model is a pairwise structured model over pose variables that characterizes local appearance properties of parts and geometric part-part interactions. The search over the full pose space is linear time in the number of parts when the part-part dependencies form a tree. However, the individual part state spaces are too large (typically hundreds of thousands of states) to allow complex appearance models to be evaluated densely. Most appearance models are therefore simple linear filters on edges, color and location Andriluka et al. (2009); Felzenszwalb and Huttenlocher (2005); Ferrari et al. (2008); Ramanan and Sminchisescu (2006).

Similarly, because of quadratic state-space complexity, part-part relationships are typically restricted to be image-independent deformation costs that allow for convolution or distance transform tricks to speed up inference Felzenszwalb and Huttenlocher (2005), see §3. A common problem in such models is poor localization of parts that have weak appearance cues or are easily confused with background clutter (accuracy for lower arms

in human figures is almost half of that for torso or head Andriluka et al. (2009)). Localizing these elusive parts requires richer models of individual part shape and joint part-part appearance, including contour continuation and segmentation cues, which are prohibitive to compute densely.

In order to enable richer appearance models, we propose to learn a cascade of pictorial structures (CPS) of increasing pose resolution which progressively filter the pose state space. Conceptually, the idea is similar to the work on cascades for face detection Fleuret and Geman (2001); Viola and Jones (2002), but the key difference is the use of structured models. Each level of the cascade at a given spatial/angular resolution refines the set of candidates from the previous level and then runs inference to determine which poses to filter out. For each part, the model selects poses with the largest *max-marginal* scores, subject to a computational budget. Unlike conventional pruning heuristics, where the possible part locations are identified using the output of a detector, models in our cascade use inference in simpler structured models to identify what to prune, taking into account global pose in filtering decisions. As a result, at the final level the CPS model has to deal with a much smaller hypothesis set which allows us to use a rich combination of features.

In addition to the traditional part detectors and geometric features, we are able to incorporate object boundary continuity and smoothness, as well as shape features, discussed in detail in §7. The former features represent mid-level and bottom-up cues, while the latter capture shape information, which is complementary to the traditional HoG-based part models. The approach is illustrated in the overview Figure 4.1. We apply the presented CPS model combined with the richer set of features on the Buffy and PASCAL Stickmen benchmark, improving the state-of-the-art on arm localization, as discussed in §IV.

We choose instead to model part configurations as a general linear MRF over pairwise and unary terms:

$$s(x, y) = \mathbf{w} \cdot \mathbf{f}(x, y) = \sum_{i \in \mathcal{V}_\Upsilon} \mathbf{w}_i \cdot \mathbf{f}_i(x, y_i) + \sum_{ij \in \mathcal{E}_\Upsilon} \mathbf{w}_{ij} \cdot \mathbf{f}_{ij}(x, y_i, y_j) \tag{4.1}$$

where $\Upsilon = (\mathcal{V}_\Upsilon, \mathcal{E}_\Upsilon)$ defines a tree structured graph of part interactions. The parameters

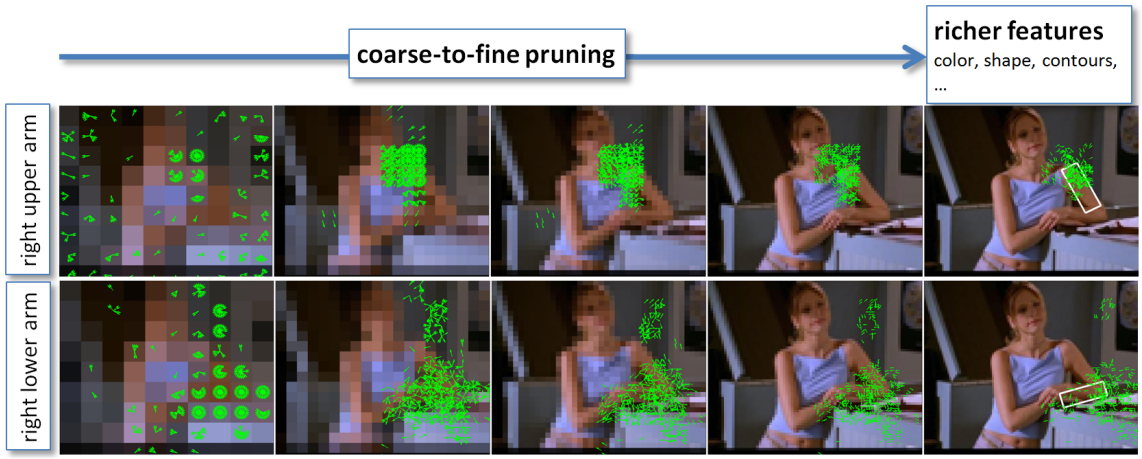

Figure 4.1: Overview of CPS: A discriminative coarse-to-fine cascade of pictorial structures filters the pose space so that expressive and computationally expensive cues can be used in the final pictorial structure. Shown are 5 levels of our coarse-to-fine cascade for the right upper and lower arm parts. Green vectors represent position and angle of unpruned states, the downsampled images correspond to the dimensions of the respective state space (but *not* the resolution at which features are computed), and the white rectangles represent classification using our final model.

of our model are the pairwise and unary weight vectors $\mathbf{w}_{ij}$ and $\mathbf{w}_i$ corresponding to the pairwise and unary feature vectors $\mathbf{f}_{ij}(x, y_i, y_j)$ and $\mathbf{f}_i(x, y_i)$. The key differences with the classical PS model are (1) our pairwise costs allow data-dependent terms, and (2) we do not constrain our parameters to fit any parametric distribution such as a Gaussian distribution, as is done in Andriluka et al. (2009); Eichner and Ferrari (2009); Felzenszwalb and Huttenlocher (2005); Ramanan and Sminchisescu (2006). This is strictly more general. For example, we can express the pairwise features used in the classical model as $y_i \cdot y_i$, $y_j \cdot y_j$, and $y_i \cdot y_j$ without requiring that their corresponding weights can be combined into a positive semi-definite covariance matrix.

In this general form (Equation 4.1), inference can *not* be performed efficiently with distance transforms or convolution as discussed in §3.1, and we rely on standard $O(nk^2)$ dynamic programming techniques to compute the best scoring assignment $\arg\max_y s(x, y)$. Many highly effective pairwise features one might design would be intractable to compute in this manner for a reasonably-sized state space—for example an 80×80 spatial grid with a part angle discretization of 24 bins yields $k^2 \approx 1$ billion part-part hypotheses.

In §4.3, we describe how we circumvent this issue via a cascade of models which aggressively prune the state space at each stage typically without discarding the correct sequence. After the state space is pruned, we are left with a small enough number of states to be able to incorporate powerful data-dependent pairwise and unary features into our model.

4.2 Related work

For unstructured, binary classification, cascades of classifiers have been quite successful for reducing computation. Fleuret and Geman (2001) propose a coarse-to-fine sequence of binary tests to detect the presence and pose of objects in an image. The learned sequence of tests is trained to minimize expected computational cost. The extremely popular Viola-Jones classifier (Viola and Jones, 2002) implements a cascade of boosting ensembles, with earlier stages using fewer features to quickly reject large portions of the state space.

Our cascade model is inspired by these binary classification cascades. In natural language parsing, several works (Carreras et al., 2008; Petrov, 2009) use a coarse-to-fine idea closely related to ours and Fleuret and Geman (2001): the marginals of a simple context free grammar or dependency model are used to prune the parse chart for a more complex grammar.

Recently, P. Felzenszwalb (2010) proposed a cascade for a structured parts-based model. Their cascade works by early stopping while evaluating individual parts, if the combined part scores are less than fixed thresholds. While the form of this cascade can

be posed in our more general framework (a cascade of models with an increasing number of parts), we differ from P. Felzenszwalb (2010) in that our pruning is based on thresholds that adapt based on inference in each test example, and we explicitly learn parameters in order to prune safely and efficiently. In Fleuret and Geman (2001); P. Felzenszwalb (2010); Viola and Jones (2002), the focus is on preserving established levels of accuracy while increasing speed. The focus in this paper is instead developing more complex models—previously infeasible due to the original intractable complexity—to improve state-of-the-art performance.

A different approach to reduce the intractable number of state hypotheses is to instead propose a small set of likely hypotheses based on bottom-up perceptual grouping principles Mori et al. (2004); Srinivasan and Shi (2007b). Mori et al. Mori et al. (2004) use bottom-up saliency cues, for example strength of supporting contours, to generate limb hypotheses. They then prune via hand-set rules based on part-pair geometry and color consistency. The shape, color and contour based features we use in our last cascade stage are inspired by such bottom-up processes. However, our cascade is solely a sequence of discriminatively-trained top-down models.

4.3 Structured Prediction Cascades

The recently introduced Structured Prediction Cascade framework Weiss and Taskar (2010) provides a principled way to prune the state space of a structured prediction problem via a sequence of increasingly complex models. There are many possible ways of defining a sequence of increasingly complex models. In Weiss and Taskar (2010) the authors introduce higher-order cliques into their models in successive stages (first unary, then pairwise, ternary, etc.). Another option is to start with simple but computationally efficient features, and add more complex features downstream as the number of states decreases. Yet another option is to geometrically coarsen the original state space and successively prune and refine. We use a coarse-to-fine state space approach with simple features until we are

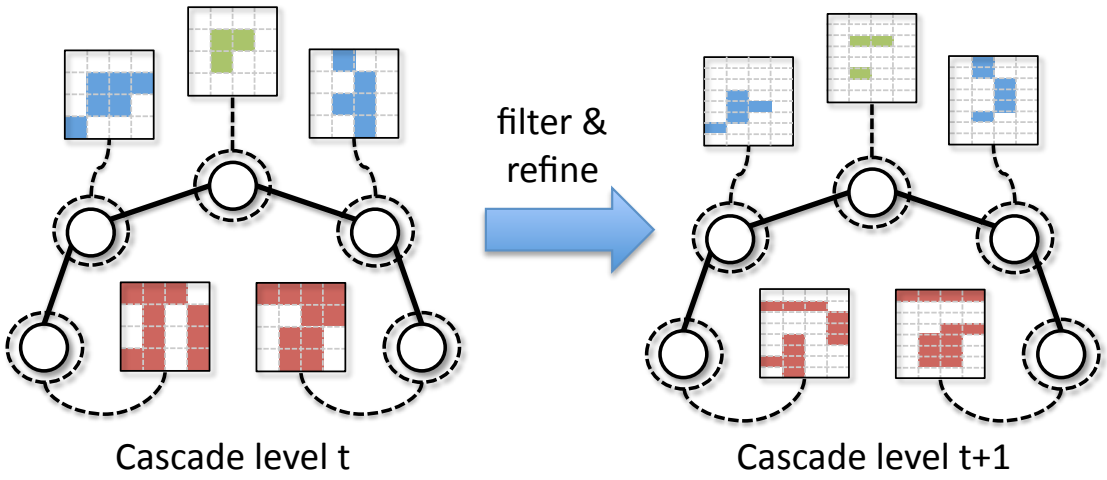

Figure 4.2: Two consecutive stages of a cascade, showing a model with a sparse set of states in a coarsened state space (top) filtering out states, and then passing them on to the next model (bottom) which works on a finer, upsampled version of the state space.

at a reasonably fine enough state space resolution and left with few enough states that we can introduce more complex features. We start with a severely coarsened state space and use standard pictorial structures unary detector scores and geometric features to perform quick exhaustive inference on the coarse state space.

4.3.1 Inference

The procedure for CPS described above is as follows. Two intermediate steps of the process are conceptualized in Figure 4.2.

- For an input x, initialize a coarse state space $\mathcal{S}_0 = \mathcal{Y}_0$ by spatially pooling states in the original space (downsampling the state space volume).
- Repeat for each cascade level $t = 0, \ldots, T-1$:
  - Run sparse, exact inference over $\mathcal{S}_t$ using the t^{th} cascade model, computing max-marginal scores.

- Filter states based on max-marginal scores to obtain $\tilde{\mathcal{S}}_t$:
  For each i, filter y_i if $s_x^\star(y_i) < t_x$, a data-dependent threshold (see §4.3.2).
- Refine the state space of $\tilde{\mathcal{S}}_t$ to obtain $\mathcal{S}_t$ for the next cascaded model.

- Predict using the final level: $y^\star = \arg\max_{y \in \mathcal{Y}_T} s(x, y)$.

The key ingredient to the cascade framework is that states are pruned using *max-marginal* scores, introduced in §2.4, computed efficiently using dynamic programming techniques. The notion of a max-marginal is intuitively explained in our model of pose estimation: The max-marginal for a part i at location y_i is the score of the highest scoring pose with part i fixed or "pinned" to location y_i. Importantly, max-marginals are a *global* quantity of a complete model of pose, rather than a local one: A part could have weak individual image evidence of being at location y_i but still have a high max-marginal score if the rest of the model believes this is a likely location.

4.3.2 Filtering threshold

When applying a cascade, we have two competing objectives that we must trade off, accuracy and efficiency: we want to minimize the number of errors incurred by each level of the cascade and maximize the number of filtered max-marginals. A natural strategy is to prune away the lowest ranked states based on max-marginal scores. Instead, we prune the states whose max-marginal score is lower than a data-specific threshold t_x: y_i is pruned if $s_x^\star(y_i) < t_x$. This threshold is defined as a convex combination of the highest score $s_x^\star = \max_y s(x, y)$ and the *mean max-marginal score*, defined as:

$$\bar{s}_x^\star = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathcal{Y}_i|} \sum_{y_i \in \mathcal{Y}_i} s_x^\star(y_i). \tag{4.2}$$

which is just the average $s_x^\star(y_i)$ over all parts and all states for each part. Our thresholding function is thus

$$t_x(s, \alpha) = \alpha s_x^\star + (1 - \alpha)\bar{s}_x^\star \tag{4.3}$$

where $\alpha \in [0, 1]$ is a parameter to be chosen that determines how aggressively to prune. When $\alpha = 1$, only the best state is kept, which is equivalent to finding the best, unconstrained assignment. When $\alpha = 0$ approximately half of the states are pruned (if the median of max-marginals is equal to the mean). The reasons for choosing $t_x(s, \alpha)$ are (1) it is a function of the image x, which allows a threshold that adapts to the difficulty of the problem, and (2) it leads to a convex learning formulation with additional guarantees, as opposed to sorting max-marginals and choosing a cutoff.

4.3.3 Learning

The goal of learning is to fit parameters to our models $\mathbf{w}$ such that they are optimized for the task of filtering states efficiently and accurately. This is notably different than standard supervised learning, which attempts to find $\mathbf{w}$ to separate the right answer from wrong. In some sense, the filtering learning object is easier to learn—the correct answer does not have to be the highest scoring, but only above the threshold value. To wit, we pose the following hard-constraint learning objective, assuming a training set of pairs $\{(x^{(j)}, y^{(j)})\}_{j=1}^{M}$:

$$\underset{\mathbf{w}}{\mathbf{minimize}} \ \frac{1}{2}||\mathbf{w}||_2^2 \tag{4.4}$$

$$\mathbf{subject\ to}: \ s(x^{(j)}, y^{(j)}) \geq t_{x^{(j)}}(s, \alpha) + 1, \quad \forall\, j \tag{4.5}$$

In words, the above is attempting to find a regularized set of weights $\mathbf{w}$ such that, in every training example, the score of the correct pose is above our image-adaptive threshold. We convert the hard constraint objective into an unconstrained hinge-loss form typical of a max-margin structured learning problem (see §??):

$$\underset{\mathbf{w}}{\mathbf{minimize}} \ \frac{\lambda}{2}||\mathbf{w}||_2^2 + \frac{1}{M}\sum_{j=1}^{M}\left[t_{x^{(j)}}(s, \alpha) - s(x^{(j)}, y^{(j)}) + 1\right]_+ \tag{4.6}$$

The learning formulation uses a simple fact about max-marginals and the definition of $t_x(s, \alpha)$ to get a handle on errors of the cascade: if $s(x, y) > t_x(s, \alpha)$, then for all i, $s_x^\star(y_i) > t_x(s, \alpha)$, so no part state of y is pruned. Given an example (x, y), this condition

$s(x, y) > t_x(s, \alpha)$ is *sufficient* to ensure that no correct part is pruned, which is our justification for using max-marginals as the pruning quality. Note that probabilistic marginals do *not* have this property: $p(y|x)$ being above a threshold does not guarantee that $p(y_i|x)$ are above a threshold for all i.

We solve (4.6) using stochastic sub-gradient descent. Given an example (x, y), we apply the following when the term $[t_x(s, \alpha) - s(x, y) + 1]_+$ (and the sub-gradient) is non-zero:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \left[-\lambda \mathbf{w} + \mathbf{f}(y, x) - \alpha \mathbf{f}(y^\star, x) - (1 - \alpha)\bar{\mathbf{f}}^\star(x)\right]. \tag{4.7}$$

Above, η is a learning rate parameter, $y^\star = \arg\max_{y'} s(x, y')$ is the highest scoring assignment and $\bar{\mathbf{f}}^\star(x)$ are the average features used by all max-marginal witnesses $y^\star(y_i)$ (explained in §2.4):

$$\bar{\mathbf{f}}^\star(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathcal{Y}_i|} \sum_{y_i \in \mathcal{Y}_i} \mathbf{f}(x, y^\star(y_i)). \tag{4.8}$$

The key distinguishing feature of this update as compared to structured perceptron is the last term which subtracts features included in all max-marginal assignments $y^\star(y_i)$.

Sequential cascade learning The stages of the cascade are learned sequentially, from coarse to fine, and each has a different set of parameters $\mathbf{w}$ and $\mathcal{Y}_i$ for each part, as well as α. The states of the next level are simply refined versions of the states that have not been pruned. We describe the refinement structure of the cascade in §IV.

4.3.4 Why not just detector-based pruning?

A naïve approach used in a variety of applications is to simply subsample states by thresholding outputs of part or sparse feature detectors, possibly combined with non-max suppression. Our approach, based on pruning on max-marginal values in a first-order model, is more sophisticated: for articulated parts-based models, strong evidence from other parts can keep a part which has weak individual evidence, and would be pruned using only detection scores. The failure of pre-filtering part locations in human pose estimation is also noted by Andriluka et al. (2009), and serves as the primary justification for their use of the

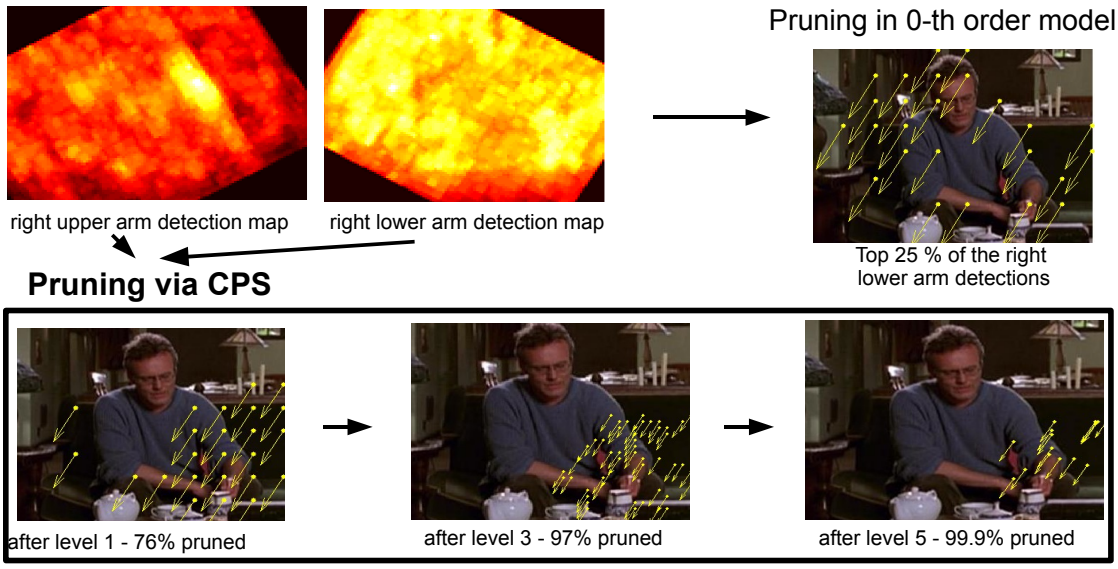

Figure 4.3: Cascade filtering example. Detector-based pruning (0th order model) by thresholding yields many hypotheses far way from the true one for the lower right arm. The CPS (bottom row), however, exploits global information (such as certainty of the shoulder location) to perform better state pruning.

dense classical PS. This is illustrated in Figure 4.3.

Chapter 5

Ensembles of Stretchable Models

5.1 Introduction

In this chapter, we focus on the task of estimating and tracking articulated 2D human pose in videos "in the wild": single-view, uncontrolled settings typical in movies, television and amateur video. Reliable parsing of human motion in such videos could vastly improve and refine action recognition and semantic retrieval. This task is made difficult by the considerable background clutter, camera movement, motion blur, poor contrast, body pose and shape variation, as well as illumination, clothing and appearance diversity. There has been an explosion of recent work on articulated pose estimation from single images of this type (Andriluka et al., 2009; Eichner and Ferrari, 2009; Felzenszwalb and Huttenlocher, 2005; Ferrari et al., 2008; Mikolajczyk et al., 2004; Ramanan, 2007; Ronfard et al., 2002), and the model proposed in §4.

Despite steady advances, localization of the most interesting parts (lower arms and hands) remains very inaccurate. Video provides additional motion cues, yet most work on articulated tracking requires manual initialization from a few frames Balan and Black (2006); Bregler and Malik (1998); Buehler et al. (2008); Ju et al. (1996b); Ren and Malik (2007); Sminchisescu and Triggs (2003). Several recent papers Ferrari et al. (2008); Lan and Huttenlocher (2005); Ramanan et al. (2005b); Sigal et al. (2004b); Weiss et al. (2010),

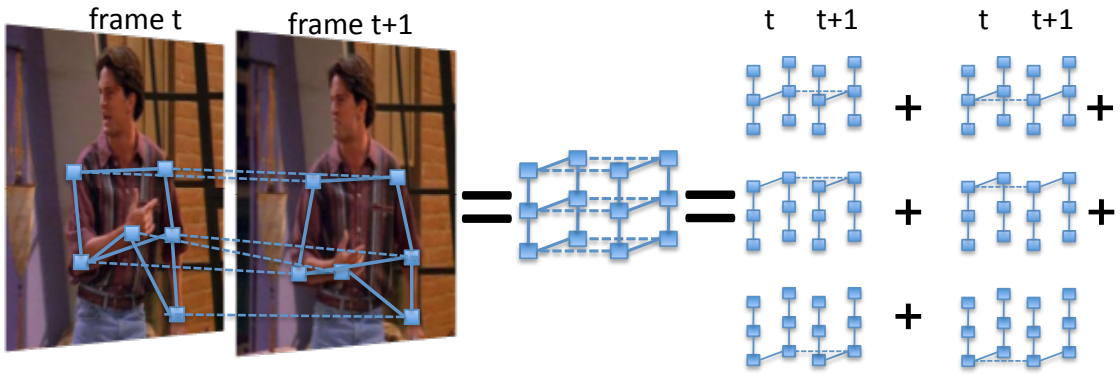

Figure 5.1: Overview. On the left, we show a full model with edges representing all pairwise relationships we want to capture within and between frames. We approximate this full, intractable loopy model by decomposing it into an ensemble of six models which cover the edge relationships of the full model. Our joint-centric representation is considerably more flexible in modeling pose variability than rigid limb-based representations Andriluka et al. (2009); Ferrari et al. (2008); Sapp et al. (2010b).

however, combine tracking and estimation without any supervised initialization; our work in this chapter follows this setting, which we call *articulated motion parsing*.

In the "strike-a-pose" parsing method of Ramanan et al. (2005b), an easily detectable canonical pose (e.g., limbs spread out) is automatically found in a long stretch of video and used as initialization for person-specific part appearance models. While this intuitive idea works well when every person strikes the easy canonical pose at least once in every video, many motion sequences are short and all the poses are difficult. Most work to date on short, difficult motions does not show significant improvement over single frame parsing Ferrari et al. (2008); Weiss et al. (2010), and sometimes causes actual degradation in accuracy when pose estimation is coupled across frames. One crucial reason for this is that joint parsing of multiple articulated parts over time involves intractable inference and learning problems, since part location/orientation variables have very large state spaces and the models are highly inter-connected (high-treewidth). Because of this barrier, previous work has resorted to approximate inference using sampling Isard and Blake (1998); Sigal

et al. (2004b); Sminchisescu and Triggs (2003); Wang et al. (2007) and variational Ferrari et al. (2008); Sigal et al. (2004a) methods, which often introduce poorly understood error and/or bias. The computational complexity of learning such models often limits the ability to learn rich features, resulting in using only simple, image-independent location-persistence coupling Ferrari et al. (2008). One notable contrast to this is the work of Ren and Malik (2007), which uses powerful image cues and inference to establish correspondences between frames and within frames. However, their method makes greedy decisions as it proceeds through frames in order to handle the intractability of maintaining a distribution of beliefs throughout the video sequence, and hence has no way of recovering from bad choices in earlier frames.

Computational considerations also usually lead to restrictive simplifying assumptions on geometry: 2D limb lengths are fixed (given global scale) Andriluka et al. (2009); Eichner and Ferrari (2009); Ferrari et al. (2008); Ramanan et al. (2005b). In typical video sequences this assumption is almost always violated because of foreshortening and body type variation, especially for the lower arms. For this paper, we collected a new challenging video dataset which extends the one in Weiss et al. (2010), which we call VideoPose2.0. In VideoPose2.0, roughly 30% of all lower arms are significantly foreshortened; see Figure 5.2 for a summary of the dataset variability.

In this work, we overcome these computational and modeling limitations using an ensemble of tractable submodels which couple locations of body joints within and across frames using rich image-dependent cues. We address the problems of foreshortening and part scale variation by using joint centers (shoulders, elbows, wrists) as parts Lee and Nevatia (2006); Rogez et al. (2008); Urtasun and Darrell (2008), instead of limbs. Each submodel in the ensemble tracks a single joint through time (e.g., left elbow), and also models the spatial arrangement of all joints in a single frame. Because of the tree structure of each submodel, we can perform efficient exact inference and use rich temporal features that depend on image appearance, e.g., color tracking and optical flow contours. The models are trained discriminatively using a large-margin loss, and we experiment with

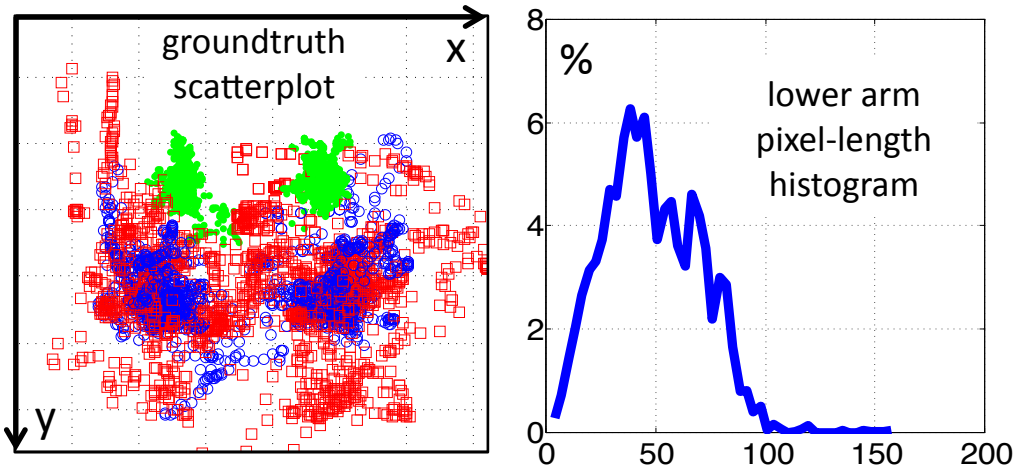

Figure 5.2: small A summary of our new video dataset, VideoPose2.0. **Left:** A scatter plot of groundtruth joints (green dot = shoulder, blue circle = elbow, red square = wrist). **Right:** Histogram of lower arm lengths in our dataset, illustrating how important it is to model relative part length in a real world setting.

a range of inference techniques that introduce increasing coupling between models (and thus increasing computational cost). Intriguingly, we find that a highly efficient method of enforcing agreement on single variables that we introduced in Weiss et al. (2010) outperforms costly approximate inference using dual decomposition.

We apply our motion parsing model on a new video dataset of highly varied and articulated poses from TV shows containing over 1,200 frames. We show significant quantitative and qualitative improvements over state-of-the-art single-frame pose estimation approaches (in particular, wrist and elbow accuracy improves by greater than 15%). In summary, the novel contributions of this work are: **(1)** an efficient ensemble of tree models for parsing human motion which covers a complex set of pairwise relationships and includes rich, image-dependent features; **(2)** stretchable 2D layout models, as opposed to the rigid parts typically used in pose estimation; **(3)** significant improvement over single frame parsing results (arguably a first in this setting) **(4)** a challenging new video

dataset, VideoPose2.0, with a high degree of motion pose variability which shows limitations of state-of-the-art methods. The VideoPose2.0 dataset, code and videos are available at *http://vision.grasp.upenn.edu/video*.

5.2 Modeling

As discussed, due to the intractability of jointly tracking multiple objects through time, previous work has made limiting assumptions in both the representation of pose (i.e., the inability to model foreshortening or fine angular granularity), and the interactions between parts, which do not capture rich, image-dependent relationships. We address each of these issues in a principled framework.

5.2.1 Stretchable models of human pose

One of the biggest limitations of current models of human pose is the "cardboard people" assumption Ju et al. (1996b): that body parts are rigid patches of fixed size (up to a global scale). In the Pictorial Structures framework, the human is represented as a collection of parts with fixed lengths, which along with the position and angle of each joint, completely determines the pose of the person [1]. Thus, wrists are always a fixed distance away from elbows in these models. This is a frequently violated assumption in realistic video, due to foreshortening and variation in body type. In the VideoPose2.0 dataset we introduce (Section IV), 30% of lower arms are significantly foreshortened, see Figure 5.2.

Two joints > one limb: Rather than model human limbs as a position and orientation, we propose a model *directly on the pixel coordinates of each joint*. Although this choice introduces more variables (one for each joint rather than for each part), the state space for each variable is drastically reduced because we no longer need to reason over a finely discretized set of angles in the state space, and we can now implicitly represent nearly any

[1] The seminal work of Felzenswalb et al. Felzenszwalb and Huttenlocher (2005) uses 10 discretized scales per part, but all modern implementations of PS use one fixed scale Andriluka et al. (2009); Ferrari et al. (2008); Sapp et al. (2010b), partly due to its prohibitive increase in the state space.

angle between parts. In current PS implementations, this is a $24\times$ reduction in the state space of each variable. In addition to this, because the length of the limb is now determined implicitly by the pixel distance between neighboring joints, the model naturally lends itself to capturing large variability in the part length. Because of its ability to represent finely discretized limb length, we refer to this model as *stretchable*, in contrast to the typical rigid, rectangle-based representation.

One side-effect of switching from a limb-centric to joint-centric model of pose is that unary attributes of the limb-centric model are now pairwise attributes of a joint-centric model. Furthermore, pairwise attributes in a limb-centric model correspond to ternary attributes in a joint-centric model, which we do not use. However, in a standard PS model, pairwise attributes are only image-independent functions of geometry, whereas in our model, pairwise potentials all incorporate image information, giving us overall a more expressive model than standard PS.

5.2.2 Ensembles of stretchable models

Ideally we want a model of human motion which captures important relationships between all correlated parts. This includes parts that are connected kinematically (e.g., left elbow, left wrist), parts that are left/right symmetric (e.g., left elbow, right elbow), and instantiations of the same part in consecutive frames (e.g., left elbow at time t, left elbow at time $t+1$). Clearly, modeling all these relationships together leads to cyclic dependencies within a single frame (due to the three symmetry edges) and between consecutive frames (due to the six tracking edges); see Figure 5.1-left.

However, in general, it is always possible to express the score of a given state assignment in a full, intractable model as the sum of scores under a set of tree sub-models that collectively cover every edge in the full model. This is the key insight that allows us to include all the rich relationships we desire: we *decompose* our model of all interesting relationships related to parsing human motion into an ensemble of submodels, all of which are trees (and therefore tractable). Each tree submodel is responsible for tracking a single

joint through time and additionally models the corresponding set of pairwise interactions between joints in a single frame (Figure 5.1-right).

Formally, we pose this problem as a structured prediction task, where the input x is a video sequence of ℓ images and the output y is a sequence of $n\ell$ variables, where n is the number of parts (joint locations) included in the model. Each output y_i is the 2D coordinate of some part joint (defined in a 80×80 discretization of the pixel space) in some frame. We also use the shortcut notation $y_{\mathbf{t}} = \{y_i \mid y_i \text{ is in frame } t\}$ to index all P joint variables in frame t.

Let $G = (\mathcal{V}, \mathcal{E})$ be our full graphical pose model; we further assume a general pairwise MRF model that decomposes over the vertices $\mathcal{V}$ and edges $\mathcal{E}$, so that

$$s(x, y) = \sum_{i \in \mathcal{V}} \mathbf{w}_i \cdot \mathbf{f}_i(x, y_i) + \sum_{(i,j) \in \mathcal{E}} \mathbf{w}_{ij} \cdot \mathbf{f}_{ij}(x, y_i, y_j). \tag{5.1}$$

Note that edges (i, j) may connect variables between consecutive frames. Let there be P tree sub-models, and $G_p = (\mathcal{V}, \mathcal{E}_p)$ be the sub-graph of G corresponding to the p'th one as in Figure 5.1-right. Then we decompose the score $s(x, y)$ into the sum of the scores of the P constituent sub-models: $s(x, y) = \sum_{p=1}^{P} s^p(x, y)$, the score of the p'th model is as in Equation 5.1, restricted to the edges $\mathcal{E}_p$, i.e.

$$s^p(x, y) = \sum_{i \in \mathcal{V}} \mathbf{w}_i^p \cdot \mathbf{f}_i(x, y_i) + \sum_{(i,j) \in \mathcal{E}_p} \mathbf{w}_{ij}^p \cdot \mathbf{f}_{ij}(x, y_i, y_j). \tag{5.2}$$

Note that we do not couple parameters across different models $\mathbf{w}^p$, so that different models can learn different parameters and effect different behaviors accorded to strengths and weaknesses dictated by their graph structures.

5.2.3 Inference

We explore several methods for combining the P independent models during test time to make a single final decision. The methods form a hierarchy of agreement criteria between the submodels: at one extreme, we enforce the constraint that all submodels must agree on the maximizing assignment to all variables, and at the other, inference is completely

decoupled across submodels. Note that there is an inherent trade-off between the degree of agreement imposed on the models and the computational cost of the corresponding inference. We present our inference methods by order of decreasing agreement below.

Full Agreement via Dual Decomposition. A natural goal is to find the argmax decoding of joint locations throughout the entire sequence of frames using our original model in Equation 5.1. However, solving the argmax decoding problem exactly is prohibitively expensive, due to the high treewidth of this cyclic graph. We use the method of Dual Decomposition (DD) Bertsekas (1999); Komodakis et al. (2007) to solve a linear programming relaxation of the decoding problem as follows. Observe that the argmax decoding problem of our full model can be decomposed into P subproblems if those problems are coupled through a global equality constraint:

$$\arg\max_{y,y^1,\ldots,y^P} \sum_{p=1}^{P} s^p(x, y^p) \text{ s.t. } y^p = y \qquad (DD) \tag{5.3}$$

Although the optimization (5.3) is still intractable because of the integral constraints, optimizing the *dual* of (5.3) is always tractable (if we first drop the integrality requirement), but no longer guaranteed to return an optimal integral solution. We solve the dual problem with sub-gradient descent. Once complete agreement between all y^p is reached, then inference has converged to the exact solution of (5.3) (although eventual convergence is not guaranteed.) In our experiments, if dual decomposition did not converge (after 500 iterations in practice, each requiring inference in all P models), we used the maximum scoring primal variables found during any descent iteration.

To solve the dual problem with sub-gradient descent, we introduce Lagrange multipliers ν^p_{ik} for every possible state assignment $y_i = k$ in every part model. We then alternate between updating the dual variables ν^p and the primal variables y^p:

$$y^p \leftarrow \arg\max_{y} \left(s^p(x, y) + \sum_{i,k} \nu^p_{ik} \mathbf{1}\left[y_i = k^p\right] \right) \tag{5.4}$$

$$\nu^p_{ik} \leftarrow \nu_{ik} - \alpha \left(\mathbf{1}\left[y^p_i = k\right] - \frac{1}{P} \sum_{p=1}^{P} \mathbf{1}\left[y^p_i = k\right] \right), \tag{5.5}$$

where α is a rate parameter chosen according to the scheme given in Komodakis et al. (2007).

Single Frame Agreement. An alternative to computing the MAP decoding of the entire sequence is to find the argmax solution while constraining model agreement to only a subset of the variables at a time. If we restrict our attention to the variables in a single time frame only, inference is considerably simpler. For each frame t, we solve

$$\arg\max_{y'_{\mathbf{t}}} \sum_{p=1}^{P} \max_{y:y_{\mathbf{t}}=y'_{\mathbf{t}}} s^p(x,y) \qquad (SF) \tag{5.6}$$

to obtain the joint configuration for that frame (remember $y_{\mathbf{t}}$ denotes the *set* of n joint variables in frame t). In words, the inner maximization finds the highest scoring sequence y subject to the constraint that the variables in frame t are fixed at positions $y'_{\mathbf{t}}$. The outer $\arg\max$ is over all possibilities of single frame configurations $y'_{\mathbf{t}}$. This extends the notion of a max-marginal of a variable (see §2.4) to a max-marginal over *a set* of variables. This method requires first computing the max-sum messages in a forward-backward message passing algorithm (see Algorithm **??**), sent to variables in frame t, in each of the submodels. Finding the argmax decoding of $y_{\mathbf{t}}$ is then equivalent to inference in a grid with n variables (in our case $n = 6$ joints). This can be solved exactly by forming cliques of size 3, for which the message passing clique-tree forms a chain. The construction is illustrated in Figure 5.3. In each clique, there are at most pairwise potentials, and since the state space of each part is relatively small ($|Y_i| \leq 500$), the $O(\sum_i |Y_i|^3)$ cost of this inference takes less than a second per frame in practice, and in our experiments took about as long as performing inference in all P tree submodels (i.e., twice as slow overall).

Single Variable Agreement. We can further narrow our subset of interest down to a single variable at a time Weiss et al. (2010). This is a weaker criteria for model agreement, but yields cheaper and simpler inference. This gives us the following inference problem

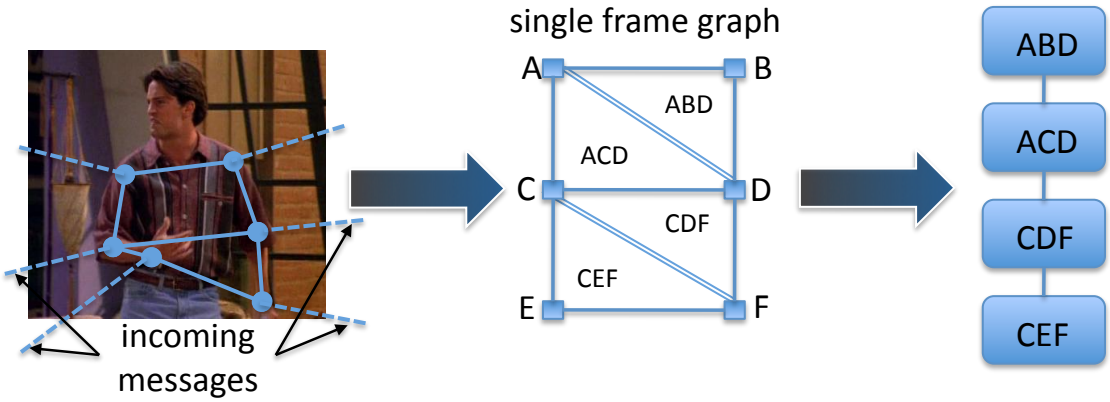

Figure 5.3: Single frame agreement construction. We first precompute messages coming into frame t from all other frames (left). We then combine 3-cliques state spaces via cartesian product to merge cliques into nodes in a larger state space (middle) to obtain a chain graph (right) over which we can perform exact inference.

for the i^{th} variable:

$$\arg\max_{y'_i} \sum_{p=1}^{P} \max_{y:y_i=y'_i} s^p(x,y) \qquad (SV) \tag{5.7}$$

This can be solved by computing max-marginals for each model using standard forward-backward message passing, summing the P max-marginal scores, and taking the highest scoring sum. Note that this is actually equal to the best assignment decoding in the full model Equation 5.1 *when all sub-models agree on the argmax*. However, this rarely occurs in practice.

Independent / No Agreement We also compared the above methods to predicting each joint using the single model in the ensemble that incorporated temporal dependencies for that specific part, which we call the $Independent$ decoding scheme.

5.2.4 Learning

Although we enforce agreement during ensemble inference at *test time*, coupling inference during training for more than one variable is prohibitively expensive to use as part of a learning procedure. Thus we learn parameters $\mathbf{w}^p$ using *decoupled* inference separately for each model. To learn parameters, we optimize a convex hinge-loss objective for each model p separately:

$$\min_{\theta_m} \frac{\lambda}{2}||\mathbf{w}^p||^2 + \frac{1}{n}\sum_{j=1}^{n}\left[\max_y s^p(x^{(j)}, y) - s^p(x^{(j)}, y^{(j)}) + 1\right]_+ \tag{5.8}$$

We optimize this problem via stochastic sub-gradient descent, and we choose λ and the number of training epochs using a held-out development set to minimize error.

Chapter 6

Locally-linear pictorial structures

6.1 Introduction

In this chapter, we focus explicitly on the multi-modal aspect of the 2D pose estimation problem. As discussed in §1.2.1, there is an enormous variation in foreground and background color and texture, and even given viewpoint and pose, the shape of body parts is highly variable due to clothing, relative scale variations, and articulation (causing foreshortening, self-occlusion and physically different body contours).

As a motivating example, consider the multitude of variations of left elbows, ignoring color and lighting variations. First, there are a range of body types, yielding infant elbows and adult elbows, male elbows and female elbows, fat elbows and skinny—a conservative estimate would say there are 5 coarse body type modes. Considering clothing, there are baggy long sleeves, tight long sleeves, stripes, solids, bare arms, and short sleeves ending near the elbow—let's assume we can summarize clothing variation around elbows with 5 appearance modes (also overly optimistic). Finally, given body type and clothing, we still must consider articulation. A typical discretization of angles to model human pose (Eichner and Ferrari, 2009; Felzenszwalb and Huttenlocher, 2005; Ramanan and Sminchisescu, 2006) uses 15° increments for a total of 24 possible relative angles. As a rough back-of-the-envelope calculation, this already amounts to $5 \times 5 \times 24 = 600$ major appearance

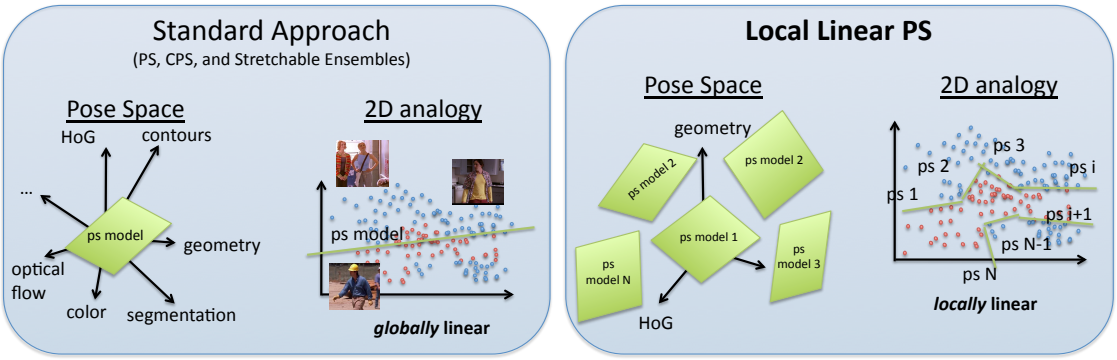

Figure 6.1: **Left:** Most pictorial structures researchers have put effort into better and larger feature spaces, in which they fit one linear model. The feature computation is expensive, and we still fail at capturing the many appearance modes in real data. **Right:** We take a different approach. Rather than introduce more and more features in hopes of high-dimensional linear separability, we model the non-linearities in simpler, lower dimensional feature spaces, using a collection of *locally* linear models.

modes, just for the elbow. See Figure 6.2 for an illustration.

Clearly, some degree of sharing between appearance modes is warranted and beneficial, in particular because it is easier to estimate fewer model parameters. However, most models developed for this problem take sharing to an extreme, and use *just a single, linear model—e.g.*, Andriluka et al. (2009); Eichner and Ferrari (2009); Ramanan and Sminchisescu (2006); Tran and Forsyth (2010) and the models developed in §4 and §5. Such models make it very difficult to capture the part variations discussed. Most researchers focus attention on improving features in hopes that a better feature space will make a linear model fit well. Only recently, several papers studied increasing the number of modes in pose models. However, these works used at most only tens of modes per part, and struggle with discovery of latent modes and/or defining an effective discrete quantization of the pose space. See §**??** for details.

Our approach: In this chapter, we propose a model of human pose that defines a mode *per training example*, to capture a local neighborhood of pose *and* appearance. This

Figure 6.2: A random sampling of 100 left elbows from the Buffy Stickmen pose dataset, removing color and intensity bias, to illustrate the huge variety of appearance due to clutter, motion blur, clothing, body type, and pose.

results in hundreds or thousands of modes in contrast to the 20 or fewer used by recent work. This gives us the ability to precisely model pose and appearance using a linear, structured model for each local neighborhood. Because of this defining characteristic, we call our model Local Linear Pictorial Structures (LLPS).

We define the extent of each neighborhood in terms of radii in appearance and pose, which decompose at a part level. As a result, the training example space is well covered, and we have overlap between modes. This allows us to heavily share training data and gives us a smoothly varying representation of the space of 2D human poses.

Using our explicit formulation of locality, no latent modes need to estimated, and no decisions made as to quantization degree and mode placement. We propose a simple convex max-margin learning objective to fit parameters of our model. We show our model is competitive with state-of-the-art approaches from this year, while employing only very simple features. Further, inference is cheap and heavily parallelizable, taking only a few seconds per image.

6.2 Related work

Here we take a particular perspective on some of the popular recent approaches Andriluka et al. (2009); Eichner and Ferrari (2009); Ramanan and Sminchisescu (2006); Sapp et al.

(2010b, 2011); Tran and Forsyth (2010). Recently, there have been two major approaches to advance the performance of pictorial structures models:

Throwing features at the problem: Equipped with a linear model of human pose, most researchers focus attention instead on increasing the quality of features, to be robust to the variety of modes discussed above—lighting-invariant texture modeling (Andriluka et al., 2009), foreground and skin color (Eichner and Ferrari, 2009; Ramanan and Sminchisescu, 2006), left-right appearance similarity (Sapp et al., 2011; Tran and Forsyth, 2010), contour and segmentation support (Sapp et al., 2010b, 2011), and more detailed description of 2D geometry (Sapp et al., 2011; Tran and Forsyth, 2010), to name a few. The hope in all of these models is that a linear model in a larger-dimensional feature space will be better able to separate the true pose configurations from false alarms, and mitigate the issue of non-linearity in lower-dimensional feature spaces, *e.g.*, using only edge orientation information.

Less features, more modes: The cost of feature computation often dominates the computation time of pose estimation, *e.g.* in (Ramanan and Sminchisescu, 2006; Sapp et al., 2010a). Instead of attempting to add more and more features into one linear, additive model there has been a recent line of work that has begun to look at modeling multiple appearance modes. Possibly the most famous example of this for rigid, non-articulated objects is the deformable parts model by Felzenszwalb et al. (Felzenszwalb et al., 2011), in which parameters for 6 different global modes (referred to as "components") are learned. Analogously in the pose estimation literature, Johnson and Everingham (S. Johnson, 2011) cluster training data based on joint locations into 16 full-body modes, and learn a pictorial structures model separately for each. Increasing refinement, Yang and Ramanan (Yang and Ramanan, 2011) introduced up to 5 modes per part (referred to as part "types") in a pictorial structures model, allowing, *e.g.*, 25 mode combinations for an elbow and wrist together. Finally, Yang et al. (Yang Wang and Liao, 2011) proposed a hierarchical model of pose, in which each sub-pose is modeled with 5-20 modes (referred to as "poselets", inspired by (Bourdev and Malik, 2009a)).

Of critical importance is how to choose modes to summarize the combined pose and appearance space of human bodies. Having many modes gives a richer description and low intra-mode variability, but is difficult to estimate given finite training sets. Having too few modes leads to high intra-mode variability and again leads to poor estimation via linear models.

When a clear definition of a mode is not available, researchers have resorted to treating mode as a discrete latent variable to be estimated during learning (Felzenszwalb et al., 2011; Yang and Ramanan, 2011). This leads to non-convex optimization, in which careful initialization is important to avoid getting stuck in bad local minima. An alternative approach has been to define modes based on geometry, typically by clustering the space of joint locations into disjoint partitions (S. Johnson, 2011; Yang Wang and Liao, 2011). This has its own issues in choosing the right level of quantization, and how to cover the space of poses correctly. Importantly, *no* human pose estimation work considers defining modes as a function of appearance, which, as discussed, is the source of much of the nonlinearity in the space of 2D human poses.

Other local modeling methods: In the machine learning literature, there is a vast array of local methods for prediction. Many of these require costly parameter estimation at test time, such as locally weighted logistic regression and KNN-SVM (Zhang et al., 2006). Although we call our method locally linear, it does not estimate parameters at test time like these techniques.

Different from those, nearest-neighbor methods are powerful, but live and die by the right choice of distance function. Standard norms fare poorly in high-dimensional spaces. Learning distance functions for nearest neighbors has achieved some success, *e.g.* Large-Margin KNN (Weinberger et al., 2006), which seeks to learn a global distance function for the whole sample space. A refinement of this is to learn local distance functions— (Frome et al., 2007) and the recent Exemplar-SVM (Malisiewicz et al., 2011) both learn distance functions per example. These works are quite similar in spirit to ours, but focus on object classification and detection, not structured, articulated part localization.

At the core of our method is a definition of a local neighborhood. Our definition uses some of the same information used to define poselets (Bourdev and Malik, 2009a), which are clusters of subsets of pose configurations. However, their model is a simple voting scheme for person detection. There is no structure between parts (poselets), and no sharing or notion of overlapping neighborhoods.

6.3 Model

We pose the problem of 2D human pose estimation as a structured prediction task. Let x represent a given input image, and y represent the location of P parts in image coordinates. Each variable y_i denotes the pixel coordinates (row, column) of part i in image x. For "parts" we choose to model joints and their midpoints (*e.g.*, left wrist, left forearm, left elbow) which allows us fine-grained encoding of foreshortening and rotation, as is done in (Sapp et al., 2011; Yang and Ramanan, 2011).

The standard Pictorial Structures model described in §3 is a linear model which decomposes into a sum of unary and pairwise linear terms. We choose a general pairwise MRF form in which the score for a part configuration specified by y in image x is:

$$s(x, y) = \sum_{i\mathcal{V}}^{n} \mathbf{w}_i \cdot \mathbf{f}_i(x, y_i) + \sum_{(i,j)\in\mathcal{E}} \mathbf{w}_{ij} \cdot \mathbf{f}_{ij}(x, y_i, y_j), \tag{6.1}$$

When the graph $G = (\mathcal{V}, \mathcal{E})$ describes a tree, we can use efficient dynamic programming techniques to infer the best scoring configuration of all parts (§2.3), $y^\star = \arg\max_{y\in\mathcal{Y}} s(x, y)$. The set $\mathcal{Y}$ denotes the entire set of possible poses, which is exponential in the number of model parts: $|\mathcal{Y}| = |\mathcal{Y}_i|^P$, where $\mathcal{Y}_i$ is the set of possible placements of part i in the image (and is the same for all i).

6.3.1 Local Linear Pictorial Structures (LLPS)

Instead of a single, linear model as in Equation 6.1, we model human pose with a collection of linear models which describe different local neighborhoods. Let us consider M such

models, which we index $m = 1 \ldots M$:

$$s^m(x, y) = \sum_{i=1}^{n} \mathbf{w}_i^m \cdot \mathbf{f}_i(x, y_i) + \sum_{(i,j)\in\mathcal{E}} \mathbf{w}_{ij}^m \cdot \mathbf{f}_{ij}(x, y_i, y_j), \tag{6.2}$$

Each model shares the same tree structure $\mathcal{E}$ and features $\mathbf{f}_i(\cdot),\ \mathbf{f}_{ij}(\cdot)$ for the sake of simplicity; it is easy to extend this to a heterogenous collection of models. The M models' parameters $\mathbf{w}^M = [\mathbf{w}_i^m;\ \mathbf{w}_{ij}^m]$ are learned to fit a local neighborhood around each training exemplar, discussed in §6.4.

The full LLPS model introduces an extra variable $z \in [1, M]$ into the state of the model to infer, at test time, both the best local model to use, and the best placement of joints given that model:

$$s(x, y, z) = s^z(x, y) \tag{6.3}$$

$$z^\star, y^\star = \underset{z\in[1,M],\ y\in\mathcal{Y}}{\arg\max}\ s(x, y, z) \tag{6.4}$$

Thus, given a test example, the test time inference procedure is straightforward (see Figure 6.3): evaluate all M local models (in practice, up to thousands of them), via max-sum inference (§2.3), saving the score and highest scoring output sequence of each. Then, we take the highest scoring of the M local models and its output sequence as a prediction of the pose. This is transparently parallelizable and fits easily into popular parallelization frameworks, such as MapReduce.

6.4 Learning

During training, we have access to a training set of images with labeled poses $S = \{(x^t, y^t)\}_{t=1}^T$. As described, we choose to model a local neighborhood in pose and appearance centered around each training example. This allows us to capture fine variations of appearance from pose, clothing, lighting, etcetera using a single linear model.

We define a neighborhood set function which maps an image and corresponding human pose to a set of indices of "close" training samples: $t \in \mathcal{N}((x, y))$ means that training

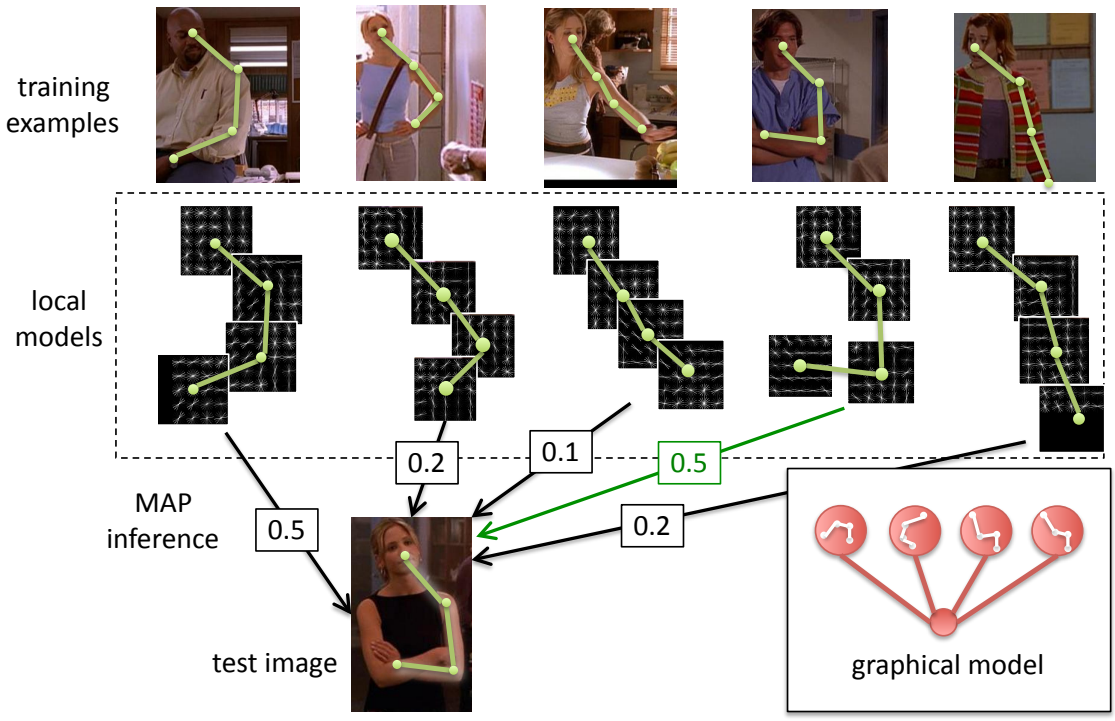

Figure 6.3: An illustration of the inference process. Each training example defines a local neighborhood model. Each model is run in parallel on a test image, and the argmax prediction is taken as a guess. Inset: the graphical model structure.

sample t is somehow "close" to (x, y) in appearance and pose. We will also overload notation to say the $\mathcal{N}(t)$ refers to the neighborhood of training sample t. The specifics of $\mathcal{N}(\cdot)$ which formalize the notions of closeness in pose and appearance are detailed in §6.5.2.

The key idea of learning is that we want to fit parameters so that each local model scores higher than other models that are not part of its local neighborhood, on the correct pose configuration. For submodel m centered about training sample (x^m, y^m), we would like the large-margin constraints:

$$s^m(x^t, y^t) \geq 1 + s^m(x^t, y), \tag{6.5}$$

$$s^m(x^t, y^t) \geq 1 + s^{m'}(x^t, y), \tag{6.6}$$

$$\forall y \in \mathcal{Y} \setminus \{y^t\},\ \forall t \in \mathcal{N}(m),\ \forall m' \notin \mathcal{N}(m)$$

In words, Equation **??** states that the score of the true configuration for local model m must

be higher than m's score for any other (wrong) configuration in example t. Equation ?? states that the score of the true configuration for m must also be higher than any score a "stranger" model $m' \notin \mathcal{N}(m)$ has for any pose configuration in example t. Adding slack and regularization on the parameters, we get a convex, large-margin structured loss jointly over all m local models

$$\min_{\{w^m\},\{\xi_{m,t}\}} \sum_{m=1}^{M} ||w^m||_2^2 + C \sum_{m=1}^{M} \sum_{t \in \mathcal{N}(m)} \left[\xi_{m,t} + \sum_{m' \notin \mathcal{N}(m)} \xi'_{m',t} \right] \tag{6.7}$$

$$\textbf{subject to: } s^m(x^t, y^t) \geq 1 + s^m(x^t, y) - \xi_{m,t} \tag{6.8}$$

$$s^m(x^t, y^t) \geq 1 + s^{m'}(x^t, y) - \xi'_{m',t} \tag{6.9}$$

$$\forall y \in \mathcal{Y} \setminus \{y^t\}, \forall m \in [1, M], \forall t \in \mathcal{N}(m), \forall m' \notin \mathcal{N}(m)$$

Note that there is in practice significant overlap in neighborhood sets rather than disjoint partitions. This gives a smoothly varying description of pose space.

6.4.1 Decomposable approximate learning

The learning objective in Equation 6.7 couples all models together to train them jointly. While this has attractive properties, it is unwieldy for large training sets. We now propose a series of modifications to the form of the model and Equation 6.7 to significantly simplify and parallelize training.

A simpler pairwise cost: Traditionally, the pairwise cost in PS models has been a simple geometric displacement cost: $f_{ij}(x, y_i, y_j) = d(y_i - y_j - \delta_{ij})$, where δ_{ij} is the mean pixel displacement vector between parts i and j, and $d(\cdot)$ is either a quadratic penalty (Felzenszwalb and Huttenlocher, 2005) or a binning function (Ramanan and Sminchisescu, 2006). These restricted forms of $d(\cdot)$ allow for fast inference message passing techniques via convolution or distance transforms. We use instead a uniform *box deformation cost* defined by $f_{ij}(x, y_i, y_j) = 0$ when $|y_i - y_j - \delta_i j| < b$, and $-\infty$ otherwise, where b specifies the size of the box in which, given y_j, y_i is free to deviate from δ_{ij} in pixel units. This yields

a new form for our local model

$$s^m(x,y) = \begin{cases} \sum_i \mathbf{w}_i^m \cdot \mathbf{f}_i(x,y_i), & |y_i - y_j - \delta_{ij}| < b, \ \ \forall (i,j) \in \mathcal{E} \\ -\infty & \text{otherwise} \end{cases} \tag{6.10}$$

This formulation has attractive properties: (1) It leads to even faster inference than a quadratic stretching cost, using a 2D min transform instead of a distance transform, (2) less parameters to learn in each local neighborhood.

Decoupling local models: Next, we drop the constraints

$$s^m(x^t,y^t) \geq 1 + s^{m'}(x^t,y), \forall m, \forall m' \notin \mathcal{N}(m), \forall t$$

which decouples model m's parameters from all other models m'. Model m no longer has to score the correct configuration y^t higher than other models' beliefs in the correct configuration. Instead, it only has to score the correct configuration higher than the exponentially many incorrect configurations in its local neighborhood of training images. This is equivalent to training a single PS model on a subset of the training data, independent of other models trained on other, overlapping subsets of the training data. The inherent problem with this is that now different local model scores may not be comparable. We address this with a calibration step described in §6.5.3.

Psuedolikelihood A final approximation is to assume, for training, that the structured functions $s^m(x,y)$ decompose into independent cliques, and can thus be trained independently. That is,

$$s^m(x,y) = \sum_i s^m(x, y_i | y_{\mathcal{E}(i)})$$

where $\mathcal{E}(i) = \{j \mid (i,j) \in \mathcal{E}\}$. This is a common trick that greatly simplifies structured learning, at the cost of approximation error. Combined with the pairwise cost described in Equation 6.10, this amounts to

$$s^m(x, y_i | y_{\mathcal{E}(i)}) = \begin{cases} \mathbf{w}_i^m \cdot \mathbf{f}_i(x,y_i), & |y_i - y_j - \delta_{ij}| < b, \ \ \forall j \in \mathcal{E}(i) \\ -\infty & \text{otherwise} \end{cases}$$

which means we can learn parameters $\mathbf{w}_i^m$ separately for each part i in each local model m.

In summary, by using a box deformation cost, decoupling the local models and using pseudolikelihood, we train part detectors individually. The structured SVM learning objective for each part i in each model m is now:

$$\min_{\mathbf{w}_i^m} ||\mathbf{w}_i^m||_2^2 + C \sum_t \xi_t \tag{6.11}$$

$$\begin{aligned} \text{subject to: } \mathbf{w}_i^m \cdot \mathbf{f}_i(x^t, y_i^t) &\geq 1 + \mathbf{w}_i^m \cdot \mathbf{f}_i(x^t, y_i) - \xi_t \\ \mathbf{w}_i^m \cdot \mathbf{f}_i(x^t, y_i^t) &\geq 1 + \mathbf{w}_i^m \cdot \mathbf{f}_i(x^{t'}, y_i) - \xi_t \\ \forall y_i \in \mathcal{Y}_i \setminus \{y_i^t\}, \forall t \in \mathcal{N}(m), \forall t' &\notin \mathcal{N}(m) \end{aligned}$$

We can optimize this using an off-the-shelf standard binary SVM solver. We handle the enormous number of constraints—$O(M \cdot |\mathcal{Y}_i|)$, the number of pixel locations in our dataset—by employing a cutting plane method, which starts from a random sampling of constraints, and iteratively finds violated constraints by mining for false positives, then re-optimizing.

6.5 Modeling human pose with LLPS

In §6.3 and §6.4, we detailed a general learning and inference framework for local linear structured models, with overlapping and decomposable neighborhood structure. We now fill in necessary details on the graph structure, neighborhood function, and features to describe how we apply this model to human pose estimation.

6.5.1 Local graph structure

In order to effectively use training data, we model only one side of a human— the nose, left shoulder, left elbow and left wrist. This allows us to re-use training data for the left

and right sides by horizontal mirroring, and also makes inference faster since we have a simple chain graph connecting each joint. We also insert midpoint nodes into the graph between the semantic joints—between the nose and shoulder (capturing neck curvature), the upper arm, and the forearm, similar to Yang and Ramanan (2011), and thus have $n = 7$ parts in each local linear structured model.

The decision to split the modeling into sides is also justified empirically. Localization accuracy for detecting the torso and head of a person in pose datasets is near-perfect, thus there is virtually no useful signal for the left side of a person to send to the right side through the torso and head, regarding beliefs of where parts should go. In other words, variables on the left and right side of the person are d-separated given the torso and head variables, which are observed almost deterministically.

6.5.2 Local Neighborhoods

As laid out in §6.4, we define a neighborhood function $\mathcal{N}(x, y)$ which takes as input the appearance x and human pose y and returns a set of training example indices which are considered part of the neighborhood of example (x, y). We allow our neighborhood function to decompose at the level of parts, so that we can mix and match training examples when we train models for different joints. For example, when learning parameters centered around example t, a different set of other examples may be used to train the elbow than are used to train the shoulder parameters. We define our part-based neighborhood as follows, leveraging both appearance and geometry:

Geometric neighborhood $\mathcal{N}_{geom}(y_i)$: Around a joint y_i, we consider two angles and two limb lengths which we use to define our geometric neighborhood: the angle that limb y_i,y_{i+1} makes with the horizontal axis, the inner angle between limbs y_{i-1},y_i and y_i,y_{i+1}, and the Euclidean length of limbs y_{i-1},y_i and y_i,y_{i+1}. Stacking these 4 geometric features into a vector g_i, we define our geometric neighborhood as

$$\mathcal{N}_{geom}(y_i) = \{\ t \text{ s.t. } |g_{ij} - g_{ij}^t| < \tau_j \text{ for } j = 1...4\ \},$$

thresholding on differences of angles and limbs between the example center y_i and all other examples y_i^t. In practice we set τ_j to be a tolerance of 5° for the two angles, and a tolerance of 6% of the height of a detected upper body for the limb length (more intuitively, approximately the palm size of a human hand).

Appearance neighborhood $\mathcal{N}_{app}(x, y_i)$: While there are many examples that are pose similar, we really wish to fit model parameters for a set of examples that *appears* similar, *e.g.*, to avoid trying to model different clothing or body types with one linear model. In light of this, we consider a patch centered around joint y_i, represented as a HoG descriptor vector h_i. We measure the similarity to other examples' HoG descriptor patches via normalized cross-correlation, and threshold to obtain our appearance neighborhood definition:

$$\mathcal{N}_{app}(x, y_i) = \{\ t \text{ s.t. } \frac{h_i}{||h_i||} \cdot \frac{h_i^t}{||h_i^t||} < \tau_h\ \},$$

where in practice we set $\tau_h = 0.2$ (normalized cross-correlation's range is $[-1, 1]$). Finally, we combine our appearance and geometry neighborhood definitions to obtain our neighborhood function $\mathcal{N}(x, y_i) = \mathcal{N}_{geom}(y_i) \cap \mathcal{N}_{app}(x, y_i)$.

6.5.3 Inter-Model Calibration

As described in §6.4, we decouple the training of our local models, which allows us to efficiently train models in parallel. The downside of this is that the bias and variance of the output of each model might vary greatly. One very overconfident local model may always dominate the prediction step. The problem of combining separately-trained models is also an issue in other recent works—S. Johnson (2011) uses multinomial logistic regression on validation data to predict which of 16 PS models to believe per test example, and Exemplar-SVM (Malisiewicz et al., 2011), lacking validation data, estimates posterior probabilities from already-seen training data. In our implementation, we perform inference for each local model on every training example—this data was seen during pseudo-likelihood training, but not in full model inference. We then estimate a linear scale and offset to map the range of scores of each local model on the training set to $[0, 1]$.

Mode selection: In an additional step, we also remove redundant or malevolent models—ones whose addition to the set of local models hurts performance. On training data, after calibration, we start with an empty set of models, and then iteratively greedily choose to add a new local model that increases the accuracy of the inlier set. If we consider each local model as a basis, this is a form of basis pursuit. In practice this results in a significantly smaller set of local models at test time, which speeds up inference. Details are in §IV.

Part III

Representations of 2D human pose

Chapter 7

Feature Sources

A major goal of this work was to introduce rich features into models of human pose. The standard approach is to represent limbs with an edge-based rigid template invariant to color, lighting, and minor deformations. In addition to this, we use color, shape and geometry.

7.1 Edges

The standard cue for pictorial structures is rigid templates based on edges (Andriluka et al., 2009; Eichner and Ferrari, 2009; Ferrari et al., 2008; Ramanan and Sminchisescu, 2006; Tran and Forsyth, 2010; Yang and Ramanan, 2011). Edges provide invariance to lighting bias and color. Robust descriptors on top of edges provide additional invariance to lighting changes and slight translations, by coarsely binning edge magnitude and orientation, possibly with local edge energy normalization. Andriluka et al. (2009) used Shape Context (Belongie et al., 2001) on top of Canny edge detection for a sparse representation which handles angular deformation well by binning radially.

In all our work, and many others, we use an unsigned histogram of gradients (HoG) descriptor, which partitions the image into a coarse uniform grid of cells, and measures the edge energy discretized into the uniform grid of spatial bins (cells) and 9 unsigned

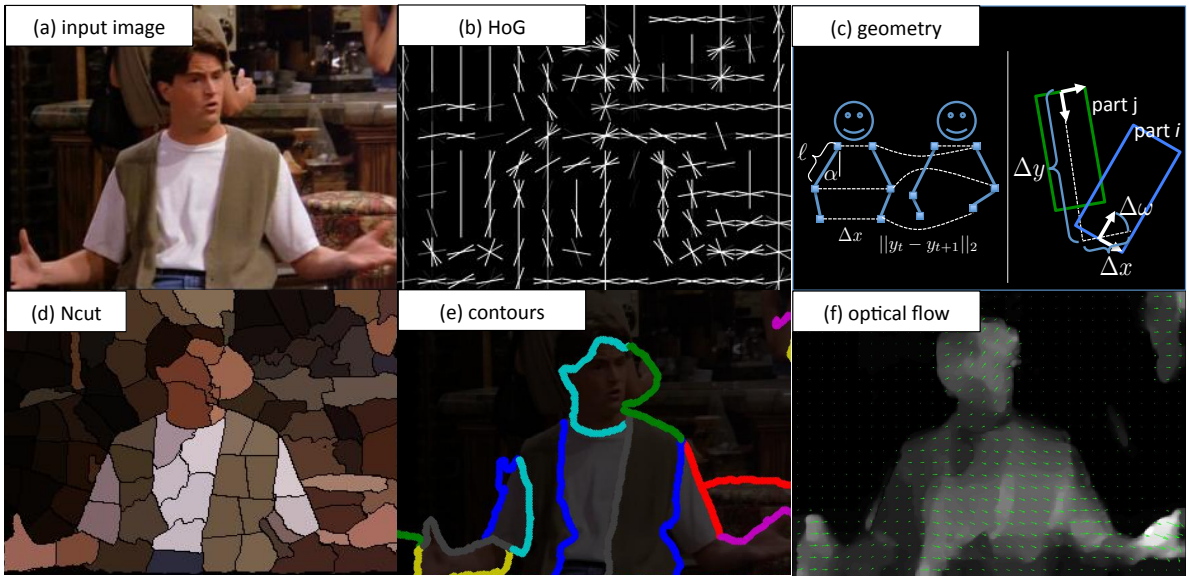

Figure 7.1: Feature sources. (a) Running example input image. (b) Coarse HoG representation. (c) Different geometric quantities we compute for our models. (d) Normalized cuts. (e) Long contours derived from Ncut segment boundaries. (f) Optical flow field shown sparsely as a quiver plot over the dense magnitude of the optical flow field at every pixel.

orientation bins. HoG was originally proposed by Dalal and Triggs (2005a) for pedestrian detection, and has been hugely popular as the representation of choice for the object recognition community as a whole, *e.g.* Felzenszwalb et al. (2008). We use a modified version of the implementation by Felzenszwalb et al. (2008), changed to use only unsigned orientations and no color. See Figure 7.1-b.

7.2 Color

Color is a tricky cue to exploit, being highly variable from image to image. However, having knowledge of the foreground and/or background color is powerful information that

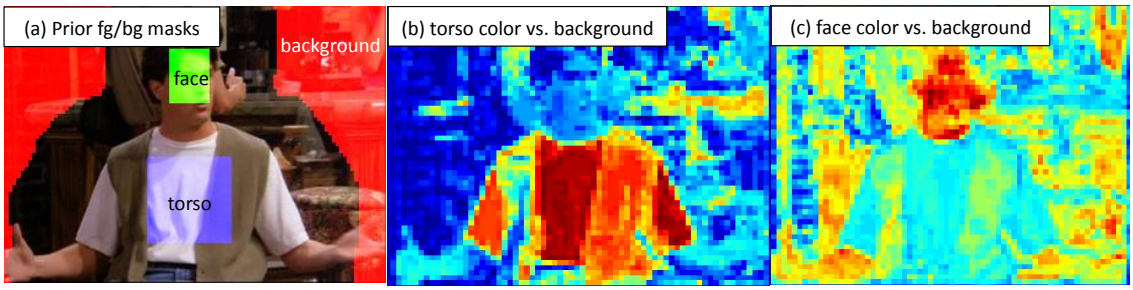

Figure 7.2: Foreground color estimation. (a) Masks obtained from groundtruth locations of face, torso and background *a priori*, overlaid on an example image. (b,c) Output of classifier learned to discriminate face versus background and torso versus background independently for every pixel.

enables us to separate unlikely poses from likely ones—we can exploit such assumptions as figure/ground color separation, and foreground constancy across the body and through time.

Foreground/Background modeling It is difficult to model the foreground or background color without first *knowing* where the foreground and background are—"a chicken and egg problem". Some works such as (Eichner and Ferrari, 2009; Ramanan and Sminchisescu, 2006) focus on this issue directly by iteratively re-estimating color and re-estimating pose.

We choose a simple and effective method inspired by Ramanan et al. (2005a) to obtain foreground skin and clothing color models in a single pre-processing step. We exploit the fact that humans are quite well localized by people detectors (Ferrari et al., 2008), and assume we have samples of pixels from the face, torso and (importantly) background. Figure 7.2 shows a mask of pixels which we sample and corresponding labels. Treating these samples as *groundtruth*, we model color using a supervised binary classifier (§2) for skin and torso color separately, using the assumed {face, torso} pixels as positive examples, and assumed background pixels as negative examples. Each pixel is its own example, classified independently.

Because we have few samples that do not completely describe the range of foreground

and background colors, and because the feature space is relatively small (the space of colors), it is quite easy to overfit a discriminative foreground/background color model to the training samples. This leads to poor generalization when we apply the learned model. Using validation data, we found that a scheme of learning the models using an Adaboost classifier composed of 10 decision trees each of maximum depth 2 (Friedman et al., 2001) worked well in practice. To complete our input foreground/background color estimation feature source, we apply the learned face and torso color classifiers to all pixels in the image independently (including the training sample pixels).

Color similarity and tracking It is also useful to *compare* the color of parts in a relative sense, for the purposes of color constancy between limbs and joints within a frame and across time. For this, we need a description of color and a notion of distances that are robust to fluctuations in lighting, hue and saturation. Because we are only concerned with local color similarity here (between patches in a single or consecutive frames), we choose a vector quantization to partition the colorspace into a coarse colormap. Specifically, we choose the method of Heckbert (1982)—`rgb2ind(·)`—which greedily chooses bisections of RGB space to minimize variance. This representation is not only robust to color fluctuations, it is also fast to compute and lends itself easily to robust histogram comparison distances, as explained below.

7.3 Shape

Shape representations have the potential to model very generic yet discriminative descriptions of objects. However, they have seen limited success in the object recognition community, typically because the representations are fragile and confounded by real-world clutter, *e.g*. Sebastian et al. (2004). Furthermore, much of the shape community is inspired by the study of early biological vision and pyschophysics, where the focuses is on *bottom-up processing*: finding objects or pieces of objects in a generic way—without an explicit model of any specific object—based on perceptual grouping principles such as

symmetry, contour continuity and parallelism, *e.g*. Biederman (1987).

In contrast, the object recognition community has relied heavily on data-driven discriminative models on top of edge template representations (Everingham et al., 2010), such as the HoG representation discussed in §7.1, and others (Dalal and Triggs, 2005a; Felzenszwalb et al., 2011; Viola and Jones, 2002). These models leverage powerful machine learning machinery to distinguish foreground from background clutter, but the rigid edge template representation is brittle to intrinsic deformations and extrinsic rotation, translation and scale.

We seek to leverage the representational power of bottom-up grouping models in our top-down model of human pose. We look at shape descriptors based on region and contour information. Recently other works have focused on using purely bottom-up shape representations in a top-down matching framework, such as Carreira et al. (2011); Toshev et al. (2010); Zhu et al. (2008). Also, some of the early work on human pose estimation was purely bottom-up driven, based on heuristics and hand-crafted rules to propose and connect limbs, *e.g*. Mori et al. (2004); Srinivasan and Shi (2007a).

We choose to use Multiscale Normalize Cut (Ncut) (Cour et al., 2005) to obtain regions. Ncut seeks a balanced partitioning of the image into meaningful regions via a spectral decomposition of a graph over image pixels, based on pairwise pixel affinities. See Figure 7.1-d.

We obtain long continuous contours in the image from the regions obtained by Ncut, as follows. We define a graph whose nodes are all boundaries between Ncut segments with edges linking touching boundaries. Each contour is a path in this graph. To reduce the number of possible paths, we restrict ourselves to all shortest paths. To quantify the smoothness of a contour, we compute an angle between each two touching segment boundaries[1]. The smoothness of a contour is quantified as the maximum angle between boundaries along this contour. Finally, we find among all shortest paths those whose length exceeds 18% of the estimated person's upper body and whose smoothness angle is

[1]The angle computed is between straight lines fit to the last third of each segment boundary

less then 45° to obtain a final set of contours, typically 15 to 30 per image. See Figure 7.1-e.

7.4 Geometry

As discussed in §3, most models of human pose consider only a unimodal function of the displacement between parts of the form $d(y_i - y_j - \delta_{ij})$, where δ_{ij} is the expected displacement between parts, and $d(\cdot)$ is typically squared Euclidean distance in a linearly transformed space. In our models we are not bound to any analytical form. Thus we make encode features based on limb lengths (Euclidean distance in pixel coordinates), relative angles between parts, absolute angles that limbs make with the horizontal, and absolute locations of limbs and joints. The "absolute" measurements are really with respect to the detected person bounding box, which can be thought of as a simple virtual "root" part which is always instantiated (observed) in our models. See Figure 7.1-c.

7.5 Motion

Motion gives us cues separate foreground from background, and fast moving parts (*e.g.*, hands) from more stationary ones (*e.g.* head). When consecutive frames of video are available, we obtain motion estimation from optical flow. Specifically, we use the fast implementation from Liu (2009a). See Figure 7.1-f.

Chapter 8

Features

There are several possible ways to organize the list of features we use in our systems - by model, by modality, by importance. We choose here to group them here by variable interactions, in keeping with the story of this thesis. To make clear which features are used in which models, we mark each feature with [CPS] (§4), [ESM] (§5), [LLPS] (§6), or some combination of the three.

8.1 Discretization

All the features described in this chapter come in a variety of units, types and ranges, both categorical and real. Typically in linear models performance suffers significantly when features are at very different scales (Fan et al., 2008). In order to incorporate these into a linear pairwise MRF model in a flexible manner, we *uniformly discretize* all the features discussed, mapping them to one of b bins. When a feature is real-valued, $b = 10$ with bin boundaries uniformly spaced between the 10^{th} and 90^{th} percentile of the feature's value over the training set. When a feature is categorical, b is equal to the number of categories. In all cases, each scalar feature gets mapped to a binary vector of length b, with exactly one entry that is 1, indicating bin membership, and the rest 0. This allows us to learn non-linear, multi-modal mappings of scalar features, as the weights of each

bin are unconstrained. The sparse representation allows us to compute $\mathbf{w}_i \cdot \mathbf{f}_i(x, y_i)$ and $\mathbf{w}_{ij} \cdot \mathbf{f}(x, y_i, y_j)$ quickly for all factors.

8.2 Limb-pair features

Color similarity [CPS:]: We measure the color similarity of kinematically-connected upper and lower arms as via χ^2-distance between histograms of quantized RGB color, described in §7.2. The histograms are accumulated within rectangles describing the limb extent. For histograms q and p, with bins indexed by p_i and q_i, the distance is defined as

$$\chi^2(p, q) = \frac{1}{2} \frac{\sum_i (p_i - q_i)^2}{\sum_i p_i + q_i}.$$

We discretize the colorspace into only 8 bins to histogram, giving us a stable but crude representation of color. The coarseness of the quantization is permissible because we only need to model a local region of a single image around a detected person.

Contour support [CPS]: We can use the set of contours $\{c_k\}$ obtained as described in §7.3 to define features for each pair of lower and upper arms, which encode the notion that those two parts should share a long smooth contour, which is parallel and close to the part boundaries. For each arm part y_i and a contour c_k we can estimate the edges of c_k which lie inside one of the halves of the supporting rectangle of y_i and whose edge normals build an angle smaller than ω with the normal of the part axis. We denote the number of those edges by $q_{ik}(\omega)$. Intuitively, a contour supports a limb if it is mostly parallel and enclosed in one of the limb sides, i.e. the value $q_{ik}(\omega)$ is large for small angles ω. A pair of arm limbs y_i, y_j should have a high score if both parts are supported by a contour c_k, which can be expressed as the following two scores

$$\text{cc}^{(1)}_{ijk}(\omega, \omega') = \frac{1}{2}\left(\frac{q_{ik}(\omega)}{h_i} + \frac{q_{jk}(\omega')}{h_j}\right) \quad \text{and} \quad \text{cc}^{(2)}_{ijk}(\omega, \omega') = \min\left\{\frac{q_{ik}(\omega)}{h_i}, \frac{q_{jk}(\omega')}{h_j}\right\}$$

where we normalize q_{ik} by the length of the limb h_i to ensure that the score is in $[0, 1]$. The first score measures the overall support of the parts, while the second measures the minimum support. Hence, for y_i, y_j we can find the highest score among all contours,

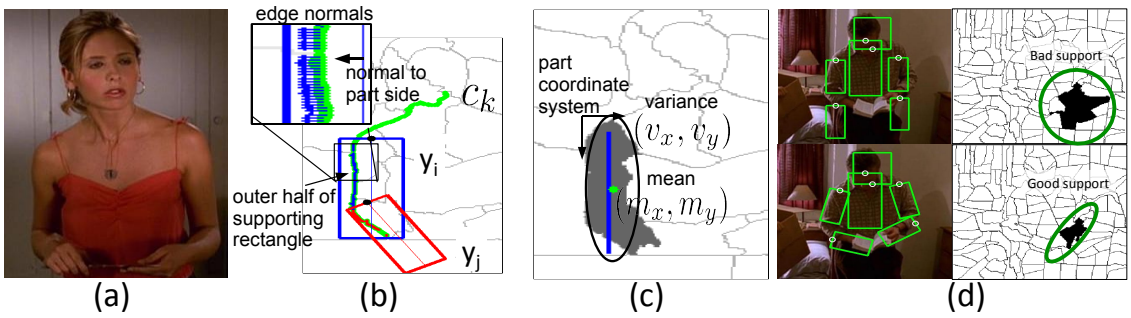

Figure 8.1: Shape features. (a) Input image. (b) Measuring a pair of limbs co-likelihood by checking attachment to long contours in the image, as described in the "Contour support" feature. (c) Measuring the likelihood of a single limb's region support, as described in the "Region support" feature. (d) Examples of right lower arm hypotheses that have poor support from underlying regions (top), and good support (bottom).

which expresses the highest degree of support which this pair of arms can receive from any of the image contours:

$$\text{cc}_{ij}^{(t)}(\omega, \omega') = \max_k \text{cc}_{ijk}^{(t)}(\omega, \omega'), \quad \text{for} \quad t \in \{1, 2\}$$

By varying the angles ω and ω' in a set of admissible angles Ω defining parallelism between the part and the contour, we obtain $|\Omega|^2$ contour features[1] The features are illustrated in Figure 8.1-(b).

Geometry [CPS]: We measure the interior angle made between parts, keeping the part identity so that the angle is in the full range $[0, 2\pi]$. We also look at difference from the expected position δ_{ij} for a pair of parts, using the signed x and y components as features.

8.3 Joint-pair features

HoG limb detectors [CPS, ESM]: For each of the 6 body parts – head, torso, upper and lower arms – we learn an individual Gentleboost classifier Friedman et al. (2000) on

[1] We set $\Omega = \{10°, 20°, 30°\}$, which results in 18 features for both scores.

HoG features (§7.1) using the Limbs Annotated in Movies Dataset[2]. We then apply these detectors densely over all orientations and locations and downsample by max-pooling or upsample via nearest-neighbor to get a heatmap of beliefs over our state space.

Region support [CPS, ESM]: We compute the first and second order moments of the segments lying under the major part axis[3] to coarsely express shape of limb hypotheses as a collection of segments R, the union of all segments that support the part hypothesis. To achieve rotation and translation invariance, we compute the moments in the part coordinate system. We include convexity information $|conv(R)|/|R|$, where $conv(\cdot)$ is the convex hull of a set of points, and $|R|$ is the number of points in the collection of segments. We also include the number of points on the convex hull, and the number of part axis points that pass through R to express continuity along the part axis. See Figure 8.1-b,c.

Foreground color [CPS, ESM]: Using likelihood heatmaps of estimated skin and torso color (§7.2), we measure the likelihood that a limb is present at a particular location by a mean heatmap value inside the rectangular extent of a part, for each color model (face and torso). This can be computed efficiently densely via convolution with an all-ones filter, separable over dimensions, for a fixed rotation. The feature can be computed efficiently over a sparse set of locations using integral images (Viola and Jones, 2002).

Geometry [CPS, ESM, LLPS]: To describe the geometry of a limb in CPS, we use as features its absolute position and orientation with respect to the image axes. Position is discretized into a coarse spatial grid, and orientation is discretized into 24 possible values; both categorical features.

In the Stretchable Ensemble Model, we use the following pairwise geometric features for limbs in a single frame: (i) length of arms, (ii) unsigned difference in angle that the upper arm makes with respect to the vertical axis, (iii) difference in x-coordinate between left-right symmetric joints, from which our model can learn a type of repulsion behavior (e.g., the left shoulder should be far from the right shoulder) as well as a left-right order of parts (e.g., the left shoulder should be to the left of the right shoulder). Note that we

[2]LAMDa is available at *http://vision.grasp.upenn.edu/video*

[3]We select segments which cover at least 25% of the part axis.

do *not* express features describing the angle of the lower arm, leaving it free to rotate and stretch. See Figure 8.2-d.

To express joint motion continuity through time in ESM, we use one simple geometric feature: the Euclidean distance between the joint in consecutive frames, discretized into 10 binary values. In LLPS, the only geometric constraint is the box distance described in §6.4.1.

Contour support [ESM]: For each arm joint-pair, we measure its support from long contours extracted in the image (§7.3) as follows: we take the number of contour points that are roughly parallel to the joint-pair line segment (angle less than 12°) and within a spatial support region (approximately 25% of the length of the average limb). We then use as a feature the number of supporting contour points, normalized by the length of the limb hypothesis. Due to the sparse contour set, this feature can be computed extremely efficiently, by quickly discarding hypotheses whose endpoints are not near any contour. See Figure 8.2-c.

Color consistency [ESM]: To capture the fact that the color of pairs of joints is often similar due to clothing and/or skin color, we describe the color consistency of pairs of joints via the χ^2-distance between color histograms obtained from small image patches (with side length 10% of image dimensions) extracted around each joint. See Figure 8.2-b.

Color tracking [ESM]: We capture the persistence of appearance over time with a simple and effective patch-based color tracker for each joint. We jointly quantize each pair of consecutive images into a small number of color indices, using minimum variance quantization (§7.2)—here with 32 colors. We then compare patches (of side length 10% of the image dimensions) around the joint in each frame using an L_0-norm (Hamming loss) distance function. Let P_t and P_{t+1} represent quantized color image patches around the joint in frame t and $t+1$, with N pixels. Then the color tracking distance feature is $||P_t - P_{t+1}||_0 = \frac{1}{N} \sum_{(r,c)} \mathbf{1} P_t(r,c) = P_{t+1}(r,c)$ where (r,c) indexes rows and columns in the patch. See Figure 8.2-b.

The coarse quantization gives us a robust way to compare the color in consecutive frames that is tolerant to changes in lighting and pixel fluctuation. The L_0 distance is more robust than Euclidean distance/radial basis similarity scores often used (*e.g.* Gould et al. (2008)), in which the distance grows with the distance in the color space and can thus be corrupted by a few large outlying pixels. Furthermore, it is a much more discriminative cue than color histogram distances across time (as used in *e.g.* Ren and Malik (2007)) because it requires that the *pixel structure* of the patch be similar in consecutive frames.

Figure from flow [ESM]: We use several features based on dense optical flow from Liu (2009b), which we compute between adjacent frames. We obtain a rough estimate of the foreground of each clip by assuming there are only 2 planes of motion: foreground (figure) and background. Given a detected person, we estimate the background motion by computing the median flow vector μ_{bg} outside the detected person bounding box, and not considering outlier flow with magnitude greater than the 75^{th} percentile. We then subtract off μ_{bg} from the flow field and take the magnitude of the flow as an estimate of foreground likelihood. We incorporate this as a limb feature by computing the average foreground likelihood sampled at 5 points evenly spaced along the line segment between joint pairs. See Figure 8.2-e.

8.4 Single joint features

Histogram of Gradients [ESM, LLPS]:

In ESM we have no specific learned model of joints, so we project down trained limb detectors from §8.3, taking a max over all possible angles and over 3 possible limb lengths (not foreshortened, somewhat foreshortened, extremely foreshortened, estimated by quantiles of dataset groundtruth limbs).

In LPPS, we represent joints with 5×5 cell Histogram of Gradients descriptors around each joint, using the implementation from Felzenszwalb et al. (2011), modified to use only 9 un-signed edge orientations. Using these features, learned parameters capture the local

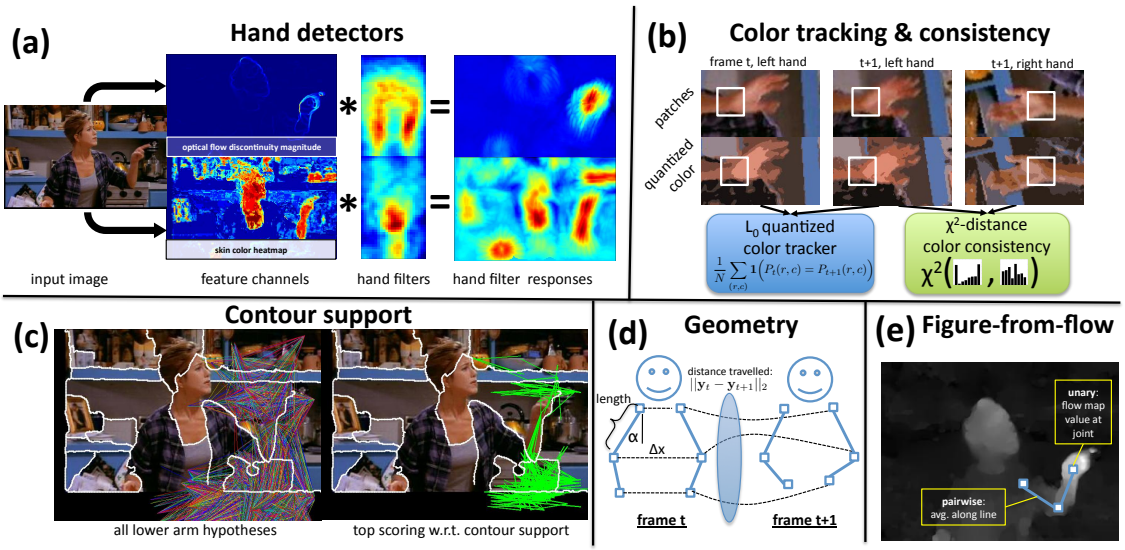

Figure 8.2: Joint and joint-pair features. See §8.3 and §8.4. (a) Hand detectors. (b) Color distances for left-right symmetry and tracking, using a quantized color space (shown). (c) Visualization of arm hypotheses that are supported by contours in the image. (d) Joint-joint geometry we employ in our Stretchable Ensemble model. (e) Figure-ground estimation by estimating two flow clusters.

shape around each joint, *specific to* the variations of pose and appearance in a small local neighborhood. We supplement the HoG channels with an extra "out-of-image" channel, to jointly learn if it's likely that a body part lies at least partially outside the image coordinates.

Color and motion-based hand detectors [ESM, LLPS]: Detecting the hand location is an extremely useful cue in anchoring all joint locations. Unfortunately, traditional template-based part detectors such as HoG detectors fail at this task due to hands' high variability in appearance, pose, and motion blur. We instead learn discriminative linear filters (Fan et al., 2008) from two different input modalities: (1) Skin detection heatmaps. (§7.2). (2) The gradient of the magnitude of the optical flow field between consecutive images (§7.5). This exploits the fact that hands are often in motion (and naturally the fastest moving body part). Both linear filters can be evaluated efficiently densely via convolution.

See Figure 8.2-a.

Figure from flow [ESM]: We again use an estimate of foreground via optical flow magnitude as in §8.3, and Figure 8.2-e. Here we report a mean foreground likelihood in a small window around the joint as a feature.

Part IV

Experiments

Chapter 9

Methodology

The main contributions of this thesis are new models of pose which allow us to incorporate new representations and features. As such, in this part we show empirically that the methods are worthwhile by comparing them to state-of-the-art methods on competitive benchmark datasets. Furthermore, we also analyze the trade-offs of the different models and the feature's effectiveness.

9.1 Datasets

We evaluate our methods on three publicly available datasets.

9.1.1 Buffy Stickmen

This dataset was first introduced in Ferrari et al. (2008). We use version 2.1 in our experiments, consisting of 748 frames from the TV show Buffy the Vampire Slayer, from episodes 2, 4, 5 and 6 from season 5. The standard protocol is to test on a given set of 235 frames that were correctly localized by an upper body detector in Ferrari et al. (2008)—within 50% overlap with the groundtruth. For training images where an upper body detector did not detect a person for which we have annotated limbs, we loosely annotated the upper body manually to simulate test time behavior.

9.1.2 PASCAL Stickmen

This dataset was first introduced by Eichner and Ferrari (2009) and contains 360 examples (version 1.0) obtained from amateur photographs culled from the PASCAL VOC 2008 challenge (Everingham et al., 2009). This is purely a test dataset; the standard protocol is to train a model on Buffy images and test on PASCAL Stickmen. We follow this protocol for CPS, but for LLPS we have different needs, discussed in §9.4.

9.1.3 VideoPose

We also apply our methods to a video dataset, which we collected ourselves: VideoPose 2.0. Clips in the dataset were hand-selected (before developing our algorithm) to highlight natural settings where state-of-the-art methods fail: a highly varied (yet realistic) range of poses, rapid gesticulation, and a significant portion of frames (30%) with foreshortened lower arms. The dataset consists of 44 short clips, 2-3 seconds in length, with a total of 1,286 frames. We use 26 clips for training, recycle 1 training clip for a development set, and use 18 for testing. As in the Buffy and PASCAL datasets, we fix global scale and translation of the person—in this case by manual annotation.

9.1.4 Discussion

The datasets Buffy, Pascal and VideoPose have different biases, as can be seen in Figure 9.1. First, note the distributions of joint locations. The least varied is Buffy, followed by Pascal and then VideoPose. In VideoPose in particular, the wrist locations have much more spread, and often lie on top of elbows, indicating more foreshortening in this dataset, and that hands are often raised up, gesticulating. Both Buffy and Pascal were collected with the 2D pictorial structure model in mind, where frames could be plucked individually to satisfy some of the 2D-based assumptions of the model. In the VideoPose dataset, we collected clips only based on duration, scale and viewpoint of the person. Note that Buffy and Pascal both have less than 20 examples with elbows raised above shoulders, whereas

VideoPose has none—this is an unusual pose in a sitcoms.

Second, observe the average images of these datasets in Figure 9.1-bottom. We can see plain biases—Buffy and VideoPose are darker (being primarily interior settings), the Buffy dataset is primarily female, and VideoPose has fewer unique background settings (the Friends' apartment, coffee shop, etc).

9.2 Evaluation Measures

Based on Problem 1, we wish to measure how well we recover "line segments describing the major anatomical parts". Ideally, we want to measure how often our guessed pose matches a groundtruth pose. However, this is nearly impossible in practice, even on the training set. It is very difficult to achieve anything near pixel-level precision—in practice, 1 pixel is approximately the size of a person's pupil at the resolution we use. This is do to uncertainty in human labeling, fixed-length limb models (in the case of classical PS and CPS), and translation invariant features which cannot guide limbs to extremely precise fits.

We consider the following relaxed measure of performance.

9.2.1 Root-Mean-Square Error (RMSE)

Because our datasets are scale normalized (to the precision of an upper-body detector), pixel distances are meaningful across examples, and semantically meaningful. As such, we could report the root-mean-square error, the average euclidean distance between predicted joints and the groundtruth joints on the test set. However, when a guess is incorrect, it can be arbitrarily far away from the groundtruth, and arbitrarily skew the RMSE.

9.2.2 Pixel error threshold

In light of the skewing problem of RMSE, we pixel accuracy within a range of thresholds. For a range of pixel distance thresholds, we report the percent of test joint guesses that

are within the threshold of the groundtruth. We explore a range of thresholds, to very precise (the semantic distance of a average palm's width in pixels), to somewhat loose (the semantic distance of an average head's height in pixels). This is illustrated in Figure 9.2.

This range of accuracies gives us a performance curve, which gives us a good idea of accuracy at different operating points. This is useful when assessing system for a range of applications—for example, it may be that for action recognition only a coarse notion of pose is required (Yang Wang and Liao, 2011), but for automatic sign language understanding (Buehler et al., 2009) we require near-certainty.

9.2.3 Percentage of Correct Parts (PCP)

Other works have proposed a measure Percentage of Correct Parts (PCP) that is a limb-based measure of correctness: a guess limb is correct if its endpoints lie within radii of the groundtruth endpoints, where the radii are half the length of the groundtruth limb (Eichner and Ferrari, 2009; Ferrari et al., 2008). In practice, the public implementation of this error measure has some peculiarities: It uses a max-norm in a coordinate space aligned to the groundtruth limb, and allows arbitrary matching of endpoints (*e.g.*, matching wrist to elbow is permissible). PCP counts a guess correct when it is perpendicular to the true limb, as long as the guess limb length is no longer than the true limb length and intersects the groundtruth limb at its midpoint.

We prefer not to use PCP because (1) its criteria for a matched guess is too relaxed (2) is discontinuous and (3) only considers one operating point instead of a range. However, for historical reasons, many works have reported results in terms of PCP, and here we do the same for compatibility.

9.3 Competitor Methods

The field of 2D human pose estimation has exploded in the last 5 years. The public datasets we report numbers on are highly competitive, with absolute performance improvements

each year for since 2008. We compare to many of the competing models. Whenever possible, we report numbers in §?? from the publicly available implementations of competitors' code; these numbers typically are different than numbers reported in papers. When public code is not available, we include PCP measures reported by the authors.

Note that the numbers reported here for our models are directly from our publicly available code, meaning any practitioner should be able to replicate results exactly. In brackets we denote how we will refer to the methods in result graphs and tables.

- Prior pose baseline [Mean]

  One reasonable sanity check is to compare with a default mean pose prior. As can be seen in Figure 9.1, there is some centrality to the data, so that guessing the mean joint position will do some fraction of joints correct. The mean pose is obtained by empirical averages of joint locations on the training sets; each joint estimated independently.

- Andriluka et al. (2009) [ARS]

  This is a classical dense PS method, as described in §3. The unary limb detectors are trained Adaboost ensembles of Shape Context on top of Canny edges. The pairwise potentials are geometric displacement, computed efficiently with distance transforms (§3.1).

- Eichner and Ferrari (2009) [EZ]

  This is a complex system built off of Ramanan and Sminchisescu (2006). The initial unary term is linear filters on image edges. It then iteratively re-parses using color estimated from the initial parse. It also uses graphcut (Boykov et al., 2001) to rule out some of the background clutter, and post-processing of probabilistic marginals to obtain final limb segments.

- Yang and Ramanan (2011) [YR]

This recent work uses HoG as its basic representation. Like our models, it is trained jointly in a discriminative learning framework. Contemporary with our proposed Stretchable Ensemble model, Yang and Ramanan (2011) also use joints as a basic unit of inference. Their work focuses on modeling a several appearance modes for each part, which they treat as latent variables to be estimated during training.

9.4 Implementation Details

Here we give various additional details about the implementation of our models, for practical purposes. All the details of the various feature implementations are provided in §7.

9.4.1 CPS

Coarse-to-Fine Cascade While our fine-level state space has size $80 \times 80 \times 24$, our first level cascade coarsens the state-space down to $10 \times 10 \times 12 = 1200$ states per part, which allows us to do exhaustive inference efficiently. We train and prune with $\alpha = 0$, effectively learning to throw away half of the states at each stage. In practice we adjust α's per part after a cascade stage is learned via cross-validation error, to prune as much as possible while retaining 95% of the groundtruth validation hypotheses.

After pruning we double one of the dimensions (first angle, then the minimum of width or height) and repeat. In the coarse-to-fine stages we only use standard PS features. HoG part detectors are run once over the original state space, and their outputs are resized to for features in coarser state spaces via max-pooling. We also use the standard relative geometric cues as described in Sec. 4. We bin the values of each feature uniformly, which adds flexibility to the standard PS modelrather than learning a mean and covariance, multimodal pairwise costs can be learned.

Final stage To obtain segments, we use NCut[19]. For the contour features we use 30 segments and for region moments 125 segments. As can be seen in §**??**, the coarse-to-fine cascade leaves us with roughly 500 hypotheses per part. For these hypotheses, we generate

all features mentioned in §7. For pairs of part hypotheses which are farther than 20% of the image dimensions from the mean connection location, features are not evaluated and an additional feature indicating this is added to the feature set. We concatenate all unary and pairwise features for part-pairs into a feature vector and learn boosting ensembles which give us our pairwise clique potentitals. This method of learning clique potentials has several advantages over stochastic subgradient learning: it is faster to train, can determine better thresholds on features than uniform binning, and can combine different features in a tree to learn complex, non-linear interactions. This approach is also a major selling point of Nowozin et al. (2011).

9.4.2 Stretchable Ensembles

Our chosen ensemble of tree models is a collection of six models that captures time persistence of each of the six joints, as well as left/right symmetric joint edges for left/right shoulder, elbows and wrists. The decomposition is shown in Figure 5.1. This covers all reasonable connections we could conceive of modeling, and allows us to incorporate all features mentioned in §7.

As input to our method, we use potential shoulder and elbow locations generated by the coarse-to-fine cascade of CPS (§4), independently for each frame. This typically yields 300-500 possible shoulder and elbow locations per image. For each of the 24 discrete elbow orientations predicted by CPS, we project possible wrist locations at 4 different lengths, chosen from the 5th, 25th, 50th, and 75th lower arm length quantiles on the training set. We then take the top 500 wrist locations scored according to the foreground color features for each frame (§7). The result is a sparse set of locations for each joint with high recall.

9.4.3 LLPS

The Buffy dataset contains 748 images, of which 235 are set aside for testing. This means we have $2 \times 513 = 1026$ half-image examples at which we center our local neighborhoods, and estimate local models. After mode selection (Section 6.5.3), we keep 291 local models to run at test time.

The Pascal dataset, on the other hand, has no training set associated with it, and people typically report cross-domain results, training on Buffy and testing on Pascal. When using models with relatively few parameters to fit, cross-domain evaluation makes little difference because training is not sensitive to fitting idiosyncrasies of a particular dataset. However, we found our model trained on Buffy to perform poorly on Pascal—Buffy is largely indoor, with a finite set of clothing styles and body types, whereas Pascal is both indoor and outdoor, and contains babies, adults and a larger variety of clothing. We feel it is a testament to the modeling power of our algorithm that such large appearance distribution changes across datasets are not handled well. As a remedy, we collected a new training set of unconstrained images similar to Pascal Stickmen from the H3D dataset Bourdev and Malik (2009a) and the VOC Challenge Everingham et al. (2009) (ensuring no overlap between this and the Pascal Stickmen test set). We selected 496 images of sufficient resolution, fully labeled and roughly upright, yielding 992 training half-images. Of these, 387 were kept after mode selection.

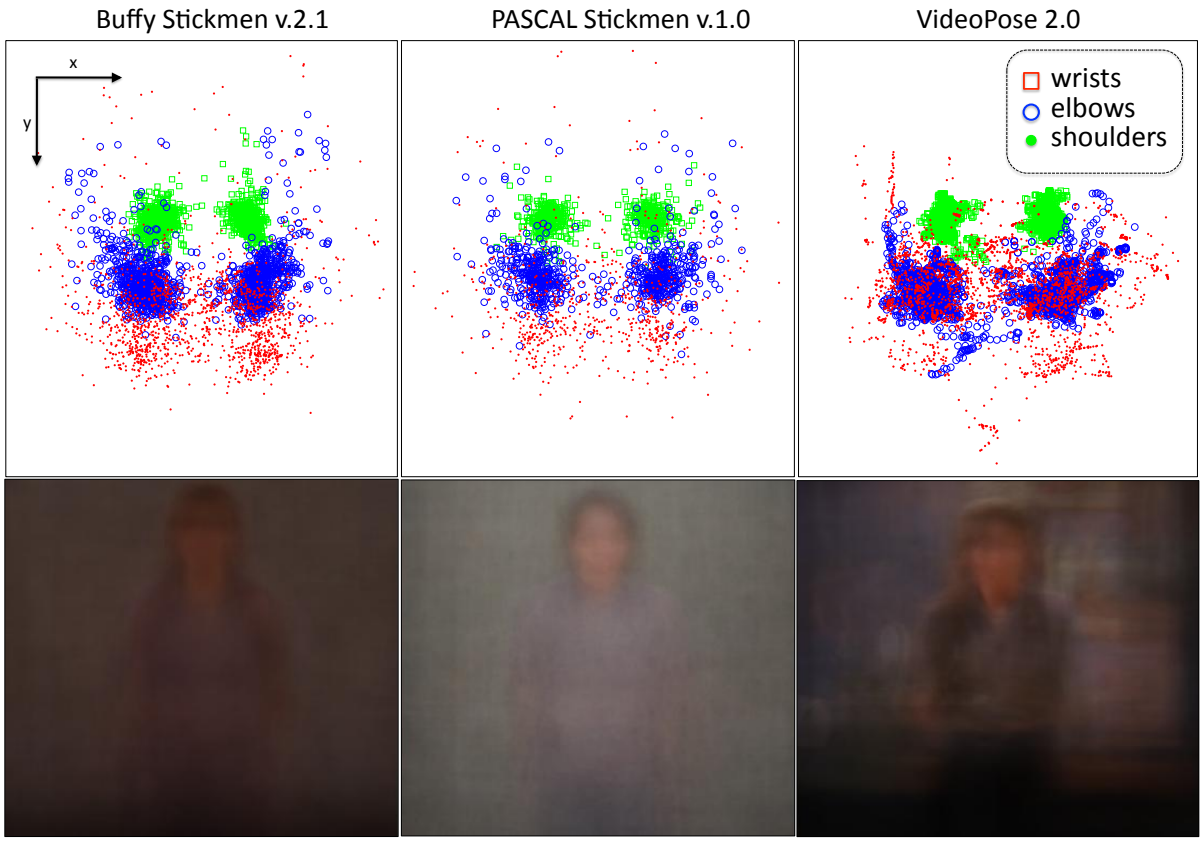

Figure 9.1: Dataset joint scatterplots and pixel averages. Here we show the spatial distribution of wrist, elbow and shoulder joints for our three datasets, both training and test, top row. Bottom row, the average images from each of the datasets.

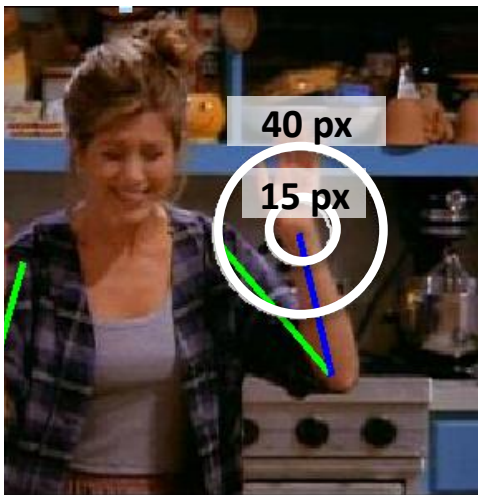

Figure 9.2: Joint error matching limits. We measure a range of pixel error distance thresholds to make a performance curve.

Chapter 10

Results

In this chapter we go through quantitative results on the datasets Buffy, Pascal and Video-Pose (§9.1), analyzing behavior and design choices of our methods. We also compare our models—CPS, ESM and LLPS—against competing models (§9.3). For much of the analysis we focus on upper and lower arms only—in particular, elbow and wrist localization accuracy. The reasons for this are that (1) torso and head localization are near-perfect given a detected person (Yang and Ramanan, 2011), (2) arms are the most interesting parts, involved in actions, hand-held objects and object-person interactions.

10.1 Coarse-to-fine cascade evaluation

In Table 10.1 we show the progress of our CPS model's coarse-to-fine cascade, in the Buffy dataset. As explained in §9.4, we start with a small state space and continue pruning and refining until we reach a somewhat fine $80 \times 80 \times 24$ grid. At the end of the cascade we are left with on average 492 states for each part, 99.67% fewer states than the complete $80 \times 80 \times 24$. We see that after 1 level the cascade, we have already pruned a large fraction of the state space away—roughly half. This is intuitive because there are many easy decisions of states to reject based on even geometry alone, *e.g.* the left elbow does not ever appear in the upper right corner of the person's bounding box.

cascade level	state dimensions	# states in the original space	# states in the pruned space	state space reduction %	near-correct lower arms unpruned (PCP)
0	10x10x12	153600	1200	00.00	100
1	10x10x24	72968	1140	52.50	76.6
3	20x20x24	6704	642	95.64	72.3
5	40x40x24	2682	671	98.25	70.5
7	80x80x24	492	492	99.67	68.4
detection pruning	80x80x24	492	492	99.67	58.6

Table 10.1: Coarse-to-fine cascade progression analysis. We show the progression of state spaces in the cascade, as well as reduction in the state space at each level (measuring efficiency), and in the last column, how many arm hypotheses remain closely matched, considering the closest match to groundtruth remaining from the unpruned hypotheses (measuring accuracy).

As the cascade progresses, we do lose arm hypotheses close to the groundtruth arms—last column of Table 10.1. However, the percent of hypotheses close to groundtruth after the cascade process is still higher than any current system's accuracy on lower arms (see Table 10.2). Thus this number (68.4%) is an upper bound on how well we could do with our small, pruned set of states.

To verify that our pruning is better than heuristic pruning, we compare to heuristic pruning in Table 10.1, last row. The heuristic here is to sample states proportional to their unary potential scores (*i.e.*, HoG limb detectors), with non-max suppression. At the same number of states sampled as we our left with via our cascade, the heuristic pruning misses 10% more lower arm hypotheses.

Finally, it could be that case that the benefits of our rich features in the last stage make discrepencies in accuracy of pruning strategies neglible. In other words, even the pruning

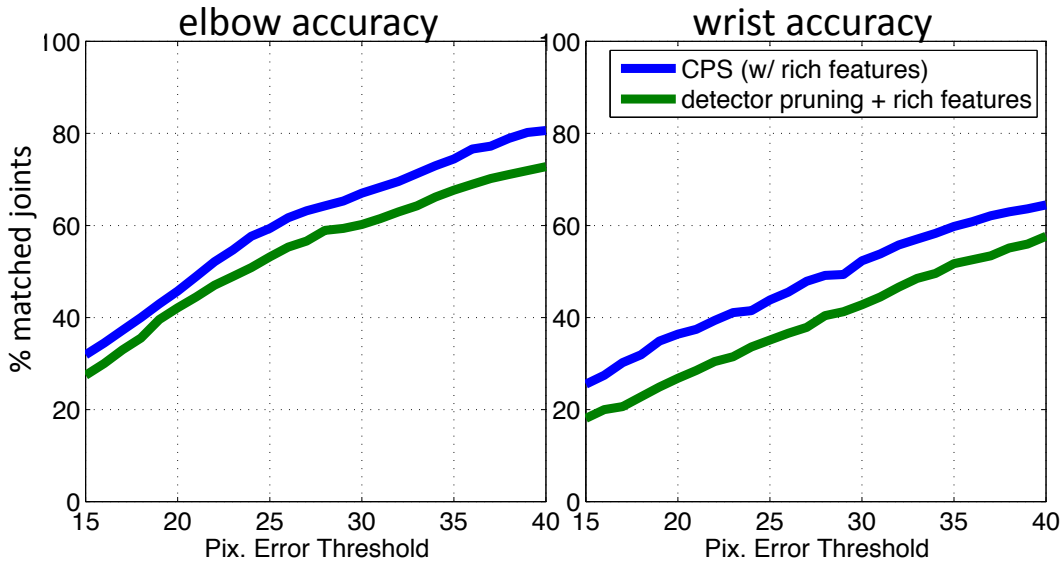

Figure 10.1: Cascade versus heuristic pruning. Here we see that our cascade progression, coupled with a rich set of features does better than heuristic pruning with the same rich features used afterwards. This indicates that not only do the rich features matter, but also the quality of the states retained from CPS.

heuristic retains 58.6% of lower arms in its hypothesis set, and it is possible that it could perform equally well at final-level prediction when using the same features as CPS. We see in Figure 10.1 that this is not this is not the case. CPS performs $5 - 10\%$ better than simple detector pruning coupled with the rich features we use in CPS. This makes a strong case that the CPS state filtering strategy is important.

10.2 Feature analysis

One of the main contributions of this thesis was technical contributions that allow us to include a veritable kitchen sink of features. At this point we wish to verify that the work done implementing, computing and learning parameters for features makes a difference in performance. This is actually quite difficult to do in general, because it is computationally

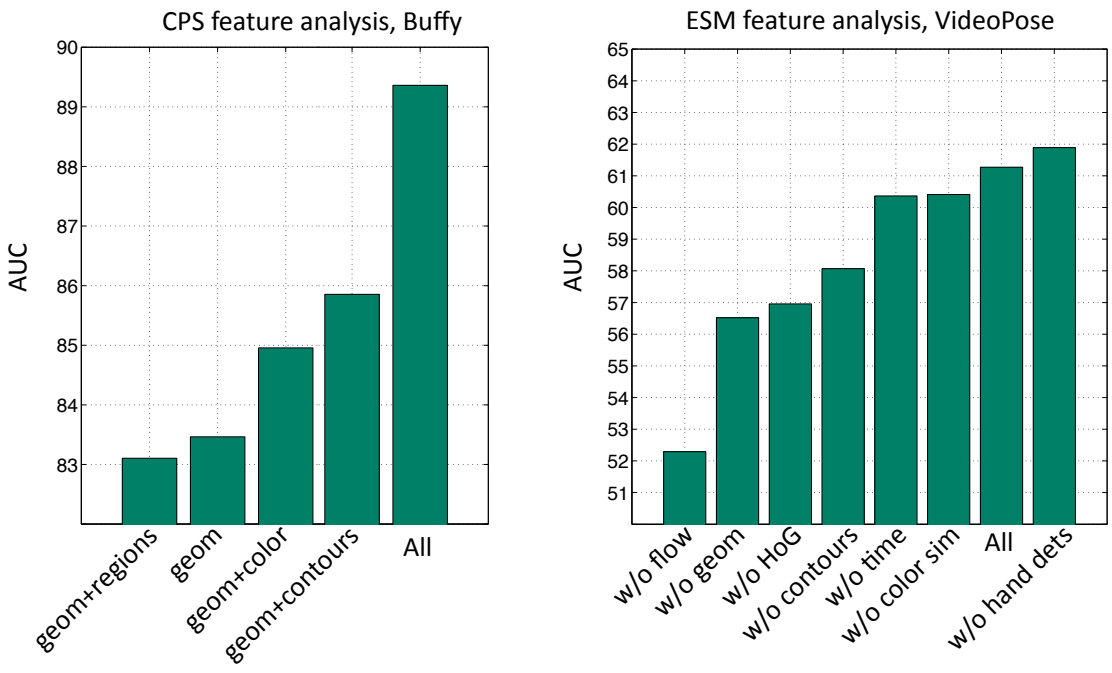

Figure 10.2: Feature analysis. On the left, we observe how adding features contributes to final system performance of CPS on Buffy, measuring the Area Under the Curve of pixel error distance. On the right, we observe how removing single feature modalities from our Ensemble of Stretchable Models affects performance.

infeasible to explore the combinatorially many possibilities of feature sets, whose interactions between modalities are significant. We analyze the importance of of features by grouping them by modality, and adding or removing them from our full systems in turn, measuring the change in performance.

In Figure 10.2 we do a feature analysis for CPS (left) and ESM (right), grouping features by coarse modalities. The first important thing to note is that, by and large, the rich features we include over standard edge template and geometry information help. For CPS, we see that including rich features of simple geometry alone helps performance, in particular pairwise cues such as color and contours that we intractable to compute without a cascade approach. An interesting except is the region based features working in isolation

PCP	Buffy v2.1		Pascal Stickmen		
method	upper arms	lower arms	upper arms	lower arms	mean
Andriluka et al. (2009)	79.3	41.2	—	—	60.3
Eichner and Ferrari (2009)	82.8	59.8	73.8	41.5	64.5
Yang and Ramanan (2011)	96.6	70.9	84.2	45.8	74.4
CPS	95.3	63.0	81.5	53.9	73.4
LPPS	90.0	62.6	83.9	47.1	70.9

Table 10.2: PCP Evaluation of single frame pose estimation. PCP is a fairly loose measure of accuracy and only reveals one precision operating point. We include the measure for historical reasons; for a more detailed picture see Figure 10.3.

actually *hurt* performance slightly on the test set. However, all features together do significantly better than any one feature modality on top of the standard geometric information.

On the right side of Figure 10.2, we do an ablative analysis of our ESM model. Interestingly, the hand detector features slightly hurt system performance on the test set, contrary to training and validation procedures. The most important individual feature modality was optical flow, which gives us a fairly good estimate of foreground/background separation in many video frames. Importantly, many of these feature modalities are not used in pose estimation models because they require joint interactions which lead to loopy, cyclic models.

10.3 System results

In this section we analayze end-to-end system results, using the publicly available code for our systems and competitors, for reproducibility's sake.

10.3.1 Single frame pose estimation

The performance of all single-frame models are shown on the Buffy and Pascal datasets in Figure **??** We see that in general, CPS and Yang and Ramanan (2011) outperform the rest, with CPS performing strictly better on Pascal. LPPS and Eichner and Ferrari (2009) are also comparable, trading off better performance in different regimes of precision. However, absolute performance localization accuracy is only one way to measure the quality of a model—see §11 for more discussion.

10.3.2 Video pose estimation

Quantitative results for pose estimation in video are shown in Figure 10.4. All three methods we explore for inference in our pose estimation video model (Ensemble of Stretchable Models) outperform the state-of-the-art single-frame methods by a significant margin. Using just a single one of our Stretchable Model trees already does significantly better than single frame models. This shows the usefulness of our stretchable model (joint-centric) representation of pose, as well as some of the rich pairwise interactions we use that other models do not. It is important to note that previous work has found the incorporating time persistence into models actually *hurt* performance (Ferrari et al., 2009; Weiss et al., 2010)—hence single frame models are the most competitive models for which to compare.

Furthermore, we explore the different agreement methods discussed in §5.2.3. From Figure 10.4, we see that the very fast, approximate decoding schemes (A single tree and SV require only a single inference pass) were comparably accurate to more expensive methods (SF takes about $2\times$ as long while we ran DD for $500\times$ iterations, each requiring inference in each submodel)[1]. We found two important trends: (1) On average, completely decoupled inference ($Independent$) was consistently about 0.75-1.5% worse than the inference methods that aggregated information across models, and (2) solving *partial*

[1]Specifically, running inference in all 6 models sequentially takes about 16 seconds per 30 frame clip (trivial to parallelize) on a Intel Xeon CPU, E5450 @ 3.00GHz. SF inference takes an additional 19 seconds.

agreement problems *exactly* (SV, SF) performed better than solving *complete* agreement *approximately* with Dual Decomposition.

The absolute accuracy values are also a testament to the difficulty of the dataset: At the tightest matching criterion, we only correctly localize half the elbows. This suggests that there is plenty of future progress to be made on this dataset, and in this domain in general.

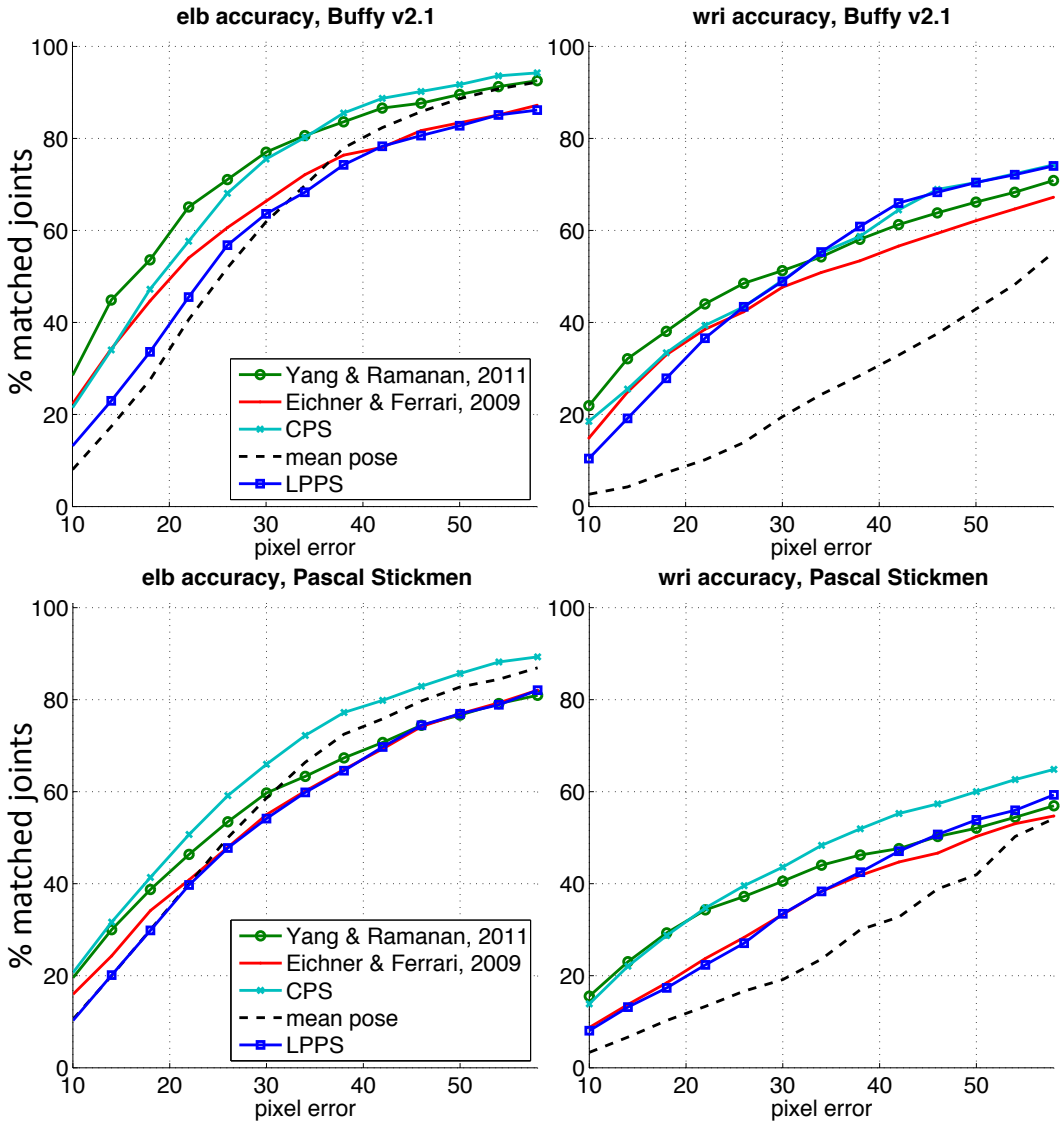

Figure 10.3: Single frame pose estimation results. Shown are our single frame models CPS and LPPS against state-of-the-art competitor models (§9.3)

.

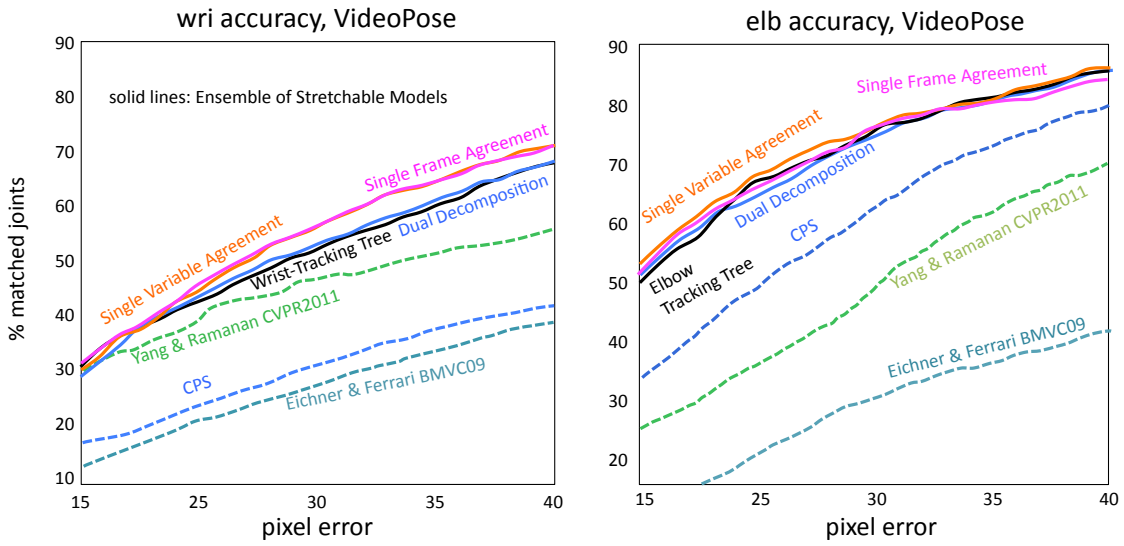

Figure 10.4: Video pose estimation results. Shown are our single frame model CPS, various forms of agreement for Ensembles of Stretchable Models, and other single frame competitors.

Part V

Discussion and Conclusions

Chapter 11

Discussion

11.1 A bug or a feature?

One of the main contributions of this thesis was technical contributions that allow us to include a veritable kitchen sink of features—the goal was to include as many feature modalities as possible in our CPS and ESM models. Having so many features makes it difficult to determine exactly what is contributing to the success of our model. From a machine learning standpoint, this is an attractive aspect of our system: given training data, we can try everything and see what works. From a computer vision standpoint, this is a disadvantage—it is difficult to gain insight into why the model is performing well exactly.

joints vs. limbs

more features vs. more modes

time vs accuracy vs complexity

what will it take to work? + more data + more computation

Chapter 12

Future work

Chapter 13

Conclusion

Fin.

Bibliography

M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Proc. CVPR*, 2009. 8, 33, 36, 37, 38, 44, 46, 47, 48, 50, 58, 59, 60, 72, 91, 101

M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 623–630. IEEE, 2010. 5

A.O. Balan and M.J. Black. An adaptive appearance model approach for model-based articulated object tracking. In *Proc. CVPR*, 2006. 46

S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *Advances in neural information processing systems*, pages 831–837, 2001. 72

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999. 53

Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987. 76

C.M. Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006. 27

Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009a. 60, 62, 94

Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision (ICCV)*, 2009b. URL http://www.eecs.berkeley.edu/~lbourdev/poselets. 32

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001. 91

C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. CVPR*, 1998. 46

P. Buehler, M. Everingham, DP Huttenlocher, and A. Zisserman. Long term arm and hand tracking for continuous sign language TV broadcasts. In *Proc. BMVC*, 2008. 46

P. Buehler, A. Zisserman, and M. Everingham. Learning sign language by watching tv (using weakly aligned subtitles). In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2961–2968. IEEE, 2009. 90

J. Carreira, F. Li, and C. Sminchisescu. Object Recognition by Sequential Figure-Ground Ranking. *International Journal of Computer Vision*, November 2011. 76

X. Carreras, M. Collins, and T. Koo. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proc. CoNLL*, pages 9–16, 2008. 39

Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In *Proc. CVPR*, 2005. 20, 76

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. Ieee, 2005a. 73, 76

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005b. 5

M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *Proc. BMVC*, 2009. 33, 38, 46, 48, 57, 58, 59, 60, 72, 74, 88, 90, 91, 101, 102

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results, 2009. 88, 94

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 76

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 78, 84

P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. Ieee, 2008. 73

P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. people.cs.uchicago.edu/pff/latent-release4/, 2011. 60, 61, 76, 83

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2006. 31

P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005. 7, 28, 31, 32, 33, 36, 38, 46, 50, 57, 65

R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proc. CVPR*, 2005. 36

V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Proc. CVPR*, 2008. 36, 46, 47, 48, 50, 72, 74, 87, 90

V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Pose search: retrieving people using their pose. In *Proc. CVPR*, 2009. 1, 102

MA Fischler and RA Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 100(22):67–92, 1973. 28, 36

G. Fleuret and D. Geman. Coarse-to-Fine Face Detection. *IJCV*, 41(1/2):85–107, 2001. 37, 39, 40

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The annals of statistics*, 28(2):337–374, 2000. 80

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001. 75

A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007. 61

Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *IJCV*, 2008. 83

Abhinav Gupta, Scott Satkin, Alexei A. Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *Computer Vision and Pattern Recognition(CVPR)*, 2011. 1

P. Heckbert. Color image quantization for frame buffer display. *Computer Graphics*, 16 (3), 1982. 75

M. Isard and A. Blake. Condensation conditional density propagation for visual tracking. *IJCV*, 1998. 47

E.T. Jaynes. Information theory and statistical mechanics. *Statistical Physics. Brandeis Lectures*, 3:160–185, 1963. 21

S. X. Ju, M.J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated motion. In *2nd Int. Conf. on Automatic Face- and Gesture-Recognition*, 1996a. 32

S.X. Ju, M.J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Automatic Face and Gesture Recognition*, 1996b. 46, 50

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. 10, 17, 23

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. ICCV*, 2007. 53, 54

X. Lan and D.P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *Proc. ICCV*, 2005. 46

M. Lee and R. Nevatia. Human pose tracking using multi-level structured models. *Proc. ECCV*, 2006. 48

Ce Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009a. 77

Ce Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, MIT, 2009b. 83

Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 61, 69

K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. *Proc. ECCV*, 2004. 46

G. Mori, X. Ren, A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *CVPR*, 2004. 40, 76

S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1668–1675. IEEE, 2011. 93

D. McAllester P. Felzenszwalb, R. Girshick. Cascade Object Detection with Deformable Part Models. In *Proc. CVPR*, 2010. 39, 40

S. Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of California at Bekeley, 2009. 39

D. Ramanan. Learning to parse images of articulated bodies. *NIPS*, 2007. 46

D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *CVPR*, pages 206–213, 2006. 36, 38, 57, 58, 59, 60, 65, 72, 74, 91

D. Ramanan, D. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Proc. CVPR*, volume 1, page 271, 2005a. 74

D. Ramanan, D.A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Proc. CVPR*, 2005b. 46, 47, 48

Xiaofeng Ren and Jitendra Malik. Tracking as repeated figure/ground segmentation. In *Proc. CVPR*, 2007. 46, 48, 83

G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P.H.S. Torr. Randomized trees for human pose detection. In *Proc. CVPR*, 2008. 48

R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *Proc. ECCV*, 2002. 46

M. Everingham S. Johnson. Learning effective human pose estimation from inaccurate annotation. In *CVPR*, 2011. 60, 61, 69

B. Sapp, C. Jordan, and B. Taskar. Adaptive pose priors for pictorial structures. In *Proc. CVPR*, 2010a. 33, 60

B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In *Proc. ECCV*, 2010b. 33, 47, 50, 59, 60

Benjamin Sapp, David Weiss, and Ben Taskar. Parsing human motion with stretchable models. In *CVPR*, 2011. 33, 60, 62

T.B. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing their shock graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):550–571, 2004. 75

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, volume 2, page 7, 2011. 2

L. Sigal, S. Bhatia, S. Roth, M.J. Black, and M. Isard. Tracking loose-limbed people. In *Proc. CVPR*, 2004a. 48

L. Sigal, M. Isard, B.H. Sigelman, and M.J. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *NIPS*, 2004b. 46, 47

C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *IJRR*, 2003. 46, 48

P. Srinivasan and J. Shi. Bottom-up recognition and parsing of the human body. *Lecture Notes in Computer Science*, 4679:153, 2007a. 76

Praveen Srinivasan and Jianbo Shi. Bottom-up recognition and parsing of the human body. In *In ICCV 05 (2005), IEEE Computer Society*, pages 824–831, 2007b. 40

Camillo J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80(10):349–363, October 2000. 32

A. Toshev, B. Taskar, and K. Daniilidis. Object detection via boundary structure segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 950–957. IEEE, 2010. 76

D. Tran and D. Forsyth. Improved Human Parsing with a Full Relational Model. *Proc. ECCV*, 2010. 58, 60, 72

K.N. Tran, I.A. Kakadiaris, and S.K. Shah. Modeling motion of body parts for action recognition. In *Jesse Hoey, Stephen McKenna and Emanuele Trucco, Proceedings of the British Machine Vision Conference, pages*, pages 64–1, 2011. 1

R. Urtasun and T. Darrell. Local Probabilistic Regression for Activity-Independent Human Pose Inference. In *Proc. CVPR*, 2008. 48

P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2):137–154, 2002. 37, 39, 40, 76, 81

J.M. Wang, D.J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *PAMI*, pages 283–298, 2007. 48

K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. Citeseer, 2006. 61

D. Weiss and B. Taskar. Structured prediction cascades. In *Proc. AISTATS*, 2010. 40

David Weiss, Benjamin Sapp, and Ben Taskar. Sidestepping intractable inference with structured ensemble cascades. In *NIPS*, 2010. 46, 47, 48, 49, 54, 102

Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *CVPR*, 2011. 60, 61, 62, 68, 72, 91, 92, 97, 101, 102

Duan Tran Yang Wang and Zicheng Liao. Learning hierarchical poselets for human parsing. In *CVPR*, 2011. 60, 61, 90

Bangpeng Yao and Li Fei-Fei. Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2012. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2012.67. 1

Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006. 61

Q. Zhu, L. Wang, Y. Wu, and J. Shi. Contour context selection for object detection: A set-to-set contour matching approach. *Computer Vision–ECCV 2008*, pages 774–787, 2008. 76