UNIVERSITY OF CALIFORNIA,
IRVINE

Tracking people and their poses

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Dennis Illyoung Park

2014

To my grandma

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am most thankful to my advisor, Deva Ramanan. I have been extremely fortunate to learn invaluable wisdom in all aspects of research; how to program, how to talk and write, how to prioritize, and mostly importantly, how to *think*. He is the most intelligent person I have ever met. He is patient enough to tolerate my slow-paced learning. He resurrected my motivation to resume research by proposing bright blueprint, whenever I was struggling with unexpected results in research experiments.

Thanks to Charless Fowlkes for sharing his keen insight. He clarifies the essence of the problem, and pinpoints crucial missing pieces in my research projects. Thanks to Alex Ihler for introducing me to machine learning through his class and providing me with opportunities to talk in seminars.

Thanks to Piotr Dollár and Larry Zitnick for the great mentoring during the summer internship at MSR and their valuable advice in pursuing research career.

Thanks to all (ex-)members of Computer Vision group at UCI; Xiangxin, Sam, Yi, Chaitanya, Hamed, Mohsen, Raul, Golnaz, Carl, Bailey, James, Phuc, Maryam, Songfan, Greg, and Julian. They made my life as a gradute student full of joy and laughter.

Thanks to all my friends, near and far, who I cannot name one-by-one. Thanks to my brother, George, and his wife, Nari. Occasional meetups with them made me feel like home.

Thanks to my parents. They always trust and support me with no condition. And thanks to Yoonnyoung, my wife, for everything.

CURRICULUM VITAE

Dennis Illyoung Park

Doctor of Philosophy in Computer Science	**2014**
University of California, Irvine	*Irvine, CA*
Master of Science in Computer Science	**2009**
University of California, Irvine	*Irvine, CA*
Bachelor of Science in Physics	**2007**
Seoul National University	*Seoul, S.Korea*

ABSTRACT OF THE DISSERTATION

Tracking people and their poses

By

Dennis Illyoung Park

Doctor of Philosophy in Computer Science

University of California, Irvine, 2014

Professor Deva Ramanan, Chair

Automatically tracking people and their body poses in unconstrained videos is a core problem of computer vision. It serves as a foundation for high-level reasoning such as activity recognition and human computer interaction. We consider two standard tracking tasks; tracking a human as its encapsulating bounding box or as an articulating body (poses).

Each task has its own challenge. The accuracy of tracking bounding boxes has been significantly improved for the past decade, but detecting small people remain challenging simply due to the lack of signals. The accuracy of tracking poses is noticeably lower, especially the one of tracking arms, mainly due to the fundamental difficulty of detecting indisticntive parts.

The algorithms for solving these problems are based on methodology of machine learning. A common pipeline is to project raw images to an invariant feature space, train a classifier (or regressor), and infer bounding boxes or poses from the trained model.

In this thesis, we aim to improve the accuracy in both tasks by proposing novel features, inference algorithms, and training schemes. In terms of tracking bounding boxes, we focus on multiresolution features and motion features that are aimed to robustly detect small people. In terms of tracking poses, we focus on combinatorial inference on part models and highly tuned appearance models.

We demonstrate our approaches using standard datasets and benchmarks of pedestrian detection and human pose estimation. Especially, our pedestrian detectors mark the top performance in Caltech Pedestrian Detection Benchmark among more than a dozen of recently developed detectors. We also achieve impressive performance in (upper body) pose estimation datasets.

Chapter 1

Overview

One of the core functions of a practical AI agent is the one to understand people in imagery. Out of all visual elements that a robot would encounter in the real world, perhaps humans are of more interest than anything else. We would expect the agent to effortlessly perform the task of identifying people, recognizing their activities and emotion, and interacting with them.

The basis of any type of such understanding is the ability to track people in unconstrained videos. The simplest type of tracking is to track a human *as a blob* by, for example, assigning a rectangular bounding (optionally with an ID) that tightly encapsulates a potential human (Fig.1.1). This minimal information already spawn high-impact applications, such as pedestrian detection systems in the automotive industry.

A more useful type of tracking is to track her *pose*, which is often represented by a set of 2D body joint coordinates, such as knees or elbows (Fig.1.1). Tracks of poses themselves serve as a succinct cues for higher level understanding such as activity recognition; humans can easily tell a “running” person from a “walking” one just by looking at a stickman animation.

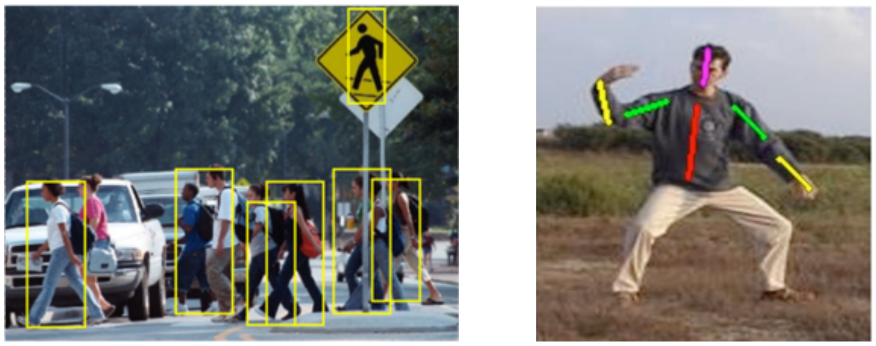


Figure 1.1: Human detection and pose estimation. The simplest type of understanding human in imagery is to detect human *as a blob*. A standard representation is a rectangular bounding box encapsulating the body (**left**). A higher level understanding is based on pose estimation, which is typically represented by 2D coordinates of body joints (**right**).

Many methods for tracking people in videos, whether as blobs or as poses, are based on static *detectors* that are independently applied to each frame of the video. The detectors are typically designed to score an image patch of human (or human pose) higher than the one of non-human (or incorrect pose). Training accurate static detectors has been a central research topic in computer vision for the past decade. Especially, vision researchers have seen impressive improvements in accuracy of *as-a-blob* human detector. Recently, the accuracy approach the level that one could consider, in order to report back a track, simply concatenating the detections from frames.

However, it remains challenging to detect small people, say, 50-pixel-tall or shorter pedestrians, while this task bears equivalent practical importance comparing with detecting larger instances (Fig.1.2). For instance, self-driving cars need to find small people so that they have enough time to stop before impact. Military surveillance cameras monitoring a large open area become much more useful, if they can identify remote intruders.

In the first part of this thesis, we focus on multi-resolution features and motion features mainly targeted to detect small people. Specifically, in **Chapter 2**, we introduce multi-resolution models that adapt their complexity to the scale of candidates. The key observation is that *scale-invariant model, which is grounded on fixed-size templates, always fails*

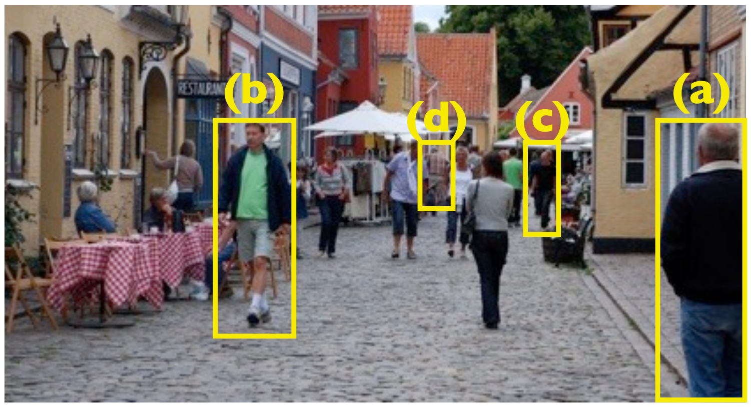


Figure 1.2: Detecting small people. In a scenario of high-speed autonomous car with rigorous standard for safety, detecting all people in this scene is equally important. In the following two chapters, we introduce multi-resolution features and motion features that are especially useful in detecting small instances such as (**c, d**).

in detecting instances in some range of scales. Our models act like a low-resolution rigid template capturing coarse contours of body given a small candidate, a high-resolution part model capturing detailed appearance and deformation given a large candidate. This work was published in *European Conference of Computer Vision, 2010* [67]. In **Chapter 3**, we introduce novel motion features that capture unique articulation motion of given object class; for example, walking motion of pedestrian. The key observation is that, *using coarse-scale optical flows, one can stabilize a sequence of 5-10 frames so that only the unique articulation motion pattern remains.* The final motion features is based on simple temporal differencing of stabilized frames. This work was published in *Conference on Computer Vision and Pattern Recognition, 2013* [69].

When it comes to tracking poses, the strategy of concatenating the detections from frames is less popular, since state-of-the-art pose detectors are much less reliable than as-a-blob detectors. The fundamental difficulty is that local appearance of parts is less distinctive than the appearance as a whole body; a patch of a lower arm looks quite similar to numerous sorts of rectangular objects. One possibility is to model spatial structures – hands must be found in a certain distance from elbows –, but those model are still inaccurate.

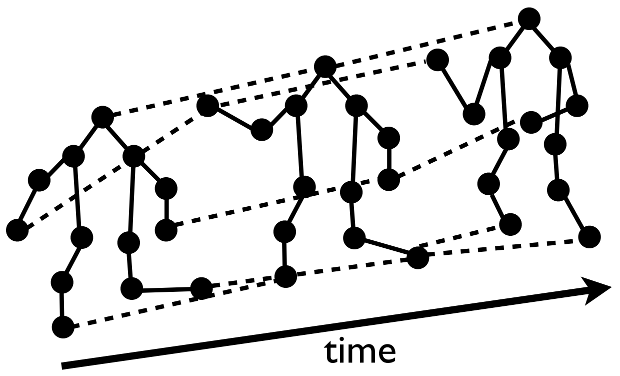


Figure 1.3: Adding temporal connectivity (**dotted**) to tree-structued spatial models (**solid**) introduce loops in the graphical model.

In terms of modeling, a natural extension is to model temporal structures as well; a hand must be found in a certain distance from an elbow, and at the same time *in the vicinity of the same hand in the previous frame*. But this causes a problem in inference: with connectivity in both spatial domain and temporal domain, the underlying graphical model becomes loopy (Fig.1.3). To avoid intractable computation involved with loopy models, an alternative method was suggested: apply a static spatial model to each of the frame to extract multiple candidates, and then use a temporal model to exhaustively search an optimal path (typically using dynamic programming). This approach is called *tracking-by-detection* framework.

In **Chapter 4**, we address the problem of extracting multiple candidate poses from a part model in tracking-by-detection framework. The rule of thumb is to generate a small number of *high-scoring and diverse pose candidates*. The problem of inferring a ranked list from graphical models have been widely explored in speech recognition community under the title of *N-best algorithms*. Naively applying those algorithms in any vision problem, however, returns a list with no diversity: *the second best pose will be a pixel-shifted version of the best pose* (Fig.1.4). We propose N-best *maximal* decoders that incorporate *non-maxima suppression cues* in the existing N-best algorithms, and that therefore guarantee a well-defined form of diversity in the list. This work was published in *International Conference on Computer Vision, 2011*, [66].

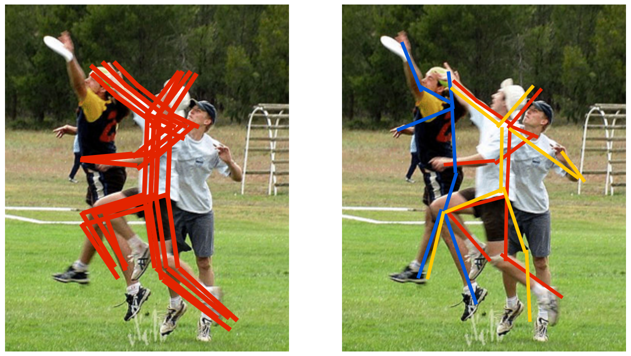


Figure 1.4: Naively applying existing N-best algorithm results in near-identical copies of the MAP estimate pose (**left**). In Chapter 4, we introduce N-best maximal decoders that produce *diverse* N-best poses (**right**).

,

Although there exists huge potential in properly modeling spatial and temporal structures, it seems also possible to circumvent solving these hard problems by training a highly-tuned appearance model. A generic pose detector is designed to accurately predict poses of a person with arbitrary appearance presented in an arbitrary background. Instead, in **Chapter 5**, we propose to use *a highly-tuned appearance models that make use of the consistency presented throughout the video*: one's clothes, skin color, body shape, as well as background scene remain mostly unchanged. We show that, given a *synthetic custom training dataset for the particular video*, we can greatly simplify the pose estimation model; as simple as using raw pixel values of the entire scene to find the nearest-neighbor from the test frame! We conclude the thesis with a short discussion.

Chapter 2

Multiresolution models for object detection

2.1 Introduction

Objects appear at a continuous range of scales in unconstrained photographs of the world. This constitutes a significant mode of intra-class variability in detection problems. The dominant perspective in the recognition community is that one should strive for scale-invariant representations, e.g., by computing features with respect to an appropriately adapted coordinate frame, as in SIFT or scanning window detectors. While this is conceptually elegant, it ignores the fact that finite sensor resolution poses an undeniable limit to scale-invariance. Recognizing a 3-pixel tall object is fundamentally harder than recognizing a 300-pixel object or a 3000-pixel object.

This is perhaps most readily apparent in common demonstrations of the importance of context in recognition (e.g., [43]). For example, the same local patch of pixels may be identified as a car or phone depending on whether the surroundings look like a street scene

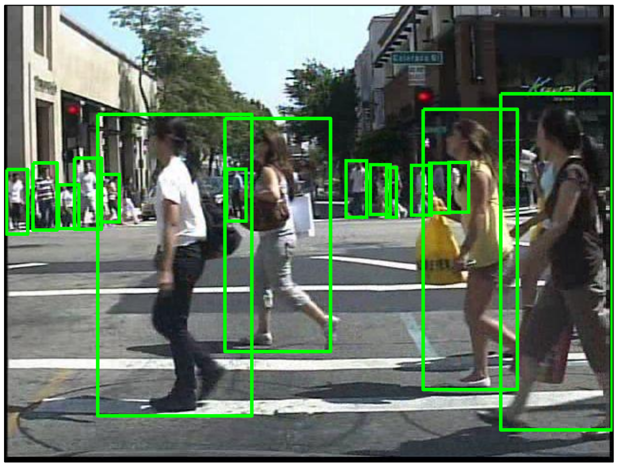


Figure 2.1: An example test image in Caltech Pedestrian dataset and its ground truth annotations. The detection results of baselines and our algorithm on this image are shown in Fig 3. Note that people appear at a wide range of scales.

or a person in an office. However, such demonstrations always involve a low-resolution, heavily-blurred image of the object in question. Given enough resolution, one *should* be able to recognize a toy-car held up to someone's ear despite the improbable context. This suggests that scene context itself should also be entered into detection in a scale-*variant* fashion with contextual cues only being used to increase the accuracy of recognizing small instances, where local image evidence is uninformative.

In this paper we propose that models for object detection should have a multiresolution structure which utilizes features ranging from detailed high-resolution parts, to whole object templates, to scene context cues. Furthermore, we treat these features in a scale dependent manner, so that high-resolution features are not used when detecting low-resolution instances.

We examine the interplay of resolution and context in the domain of pedestrian detection for autonomous vehicle navigation. Much of the recent successful work is based on template detection. We begin by asking a simple question - *what should the size of the template be*? On one hand, we want a small template that can detect small people, important for providing time for a vehicle to react. On the other hand, we want a large template that can exploit detailed features (of say, faces) to increase accuracy. Such questions are complicated by the fact a simple rigid template is not likely to accurately model both extremes, and that contextual cues should perhaps be overridden by high-confidence, large-scale detections. Using a well-known pedestrian benchmark [22], we demonstrate that contextual multiresolution models provide a significant improvement over the collective recent history of pedestrian models (as surveyed in [22]).

2.2 Related work

There is storied tradition of advocating scale-invariance in visual recognition, from scale-invariant feature detectors [55, 56, 61] to scale-invariant object representations [35, 24]. Unfortunately, such scale-invariant representations don't leverage additional pixel resolution for detecting large-scale instances.

Another family of representations deal with *multiscale* models that compute features at multiple scales. Such models are typically not multiresolution in that they do not adapt in complexity to the size of a putative detection. Examples include multiscale edge models [59] and object representations based on multiscale wavelets [65, 77]. Our approach is most similar to the multiscale part model of [33] that defines both a low-resolution root template and high-resolution part filters. We extend the publically-available code to encode adaptive multiresolution models that act as rigid templates when scoring small-scale instances and flexible part-based models when scoring large-scale instances.

There is a quite large literature on pedestrian detection, dating back to the early scanning-window classifers of [65, 37, 13]. We refer the reader to the recent surveys [22, 27] for an overview of contemporary approaches. Recent work has focused on models for handling pose variation [62, 33, 87, 94, 54], reducing complexity of learning [58, 78], and multicue combination [96, 40]. Due to its practical importance, more recent trend is to evaluate pedestrian detection in terms of both accuracy and running time [8], [19]. To the best of our knowledge, there has been no previous work on multiresolution representations of pedestrians before our work. Our work triggered attention to multiresolution, and was followed by [98].

2.3 Multiresolution Models

We will describe a family of multiresolution template models of increasing complexity. To establish notation, we begin with a description of a simple fixed-resolution template.

2.3.1 Fixed-resolution models

Let x denote an image window and $\Phi(x)$ denote its extracted features - say, histogram of oriented gradient (HOG) features [13]. Following an established line of work on scanning-window linear classifiers [34, 13], we label x as a pedestrian if

$$f(x) > 0 \qquad \text{where} \qquad f(x) = w \cdot \Phi(x) \tag{2.1}$$

Such representations are trained with positive and negative examples of pedestrian windows – formally, a set of pairs (x_i, y_i) where $y_i \in \{-1, 1\}$. Popular training algorithms include SVMs [34, 13] and boosting [18, 75]. In our work, we will train w using a linear SVM:

$$w^* = \arg\min_w \frac{1}{2} w \cdot w + C \sum_i \max(0, 1 - y_i w \cdot \Phi(x_i)) \tag{2.2}$$

One hidden assumption in such formalisms is that both the training and test data x_i is assumed to be scaled to a canonical size. For example, in Dalal and Triggs' [13] well-known detector, all training and test windows are scaled to be of size 128×64 pixels. The detector is used to find larger instances of pedestrians by scaling down in the image, implemented through an image pyramid. Formally speaking, the detector cannot be used to find instances smaller than 128×64. In practice, a common heuristic is to upsample smaller windows via interpolation, but this introduces artifacts which hurt performance [33, 22].

In this paper, we define a feature representation $\Phi(x)$ that directly processes windows of varying size, allowing one to extract additional features (and hence build a more accurate model) when x is a large-size window.

2.3.2 Multiple fixed-resolution models

Arguably the simplest method of dealing with windows of varying sizes is to build a separate model for each size. Assume that every window x arrives with a bit s that specifies whether it is "small" or "large". One can still write two templates as a single classifier $f(x, s) = w \cdot \Phi(x, s)$ where:

$$\Phi(x,s) = \begin{bmatrix} \phi_0(x) \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{if} \quad s = 0 \qquad \text{and} \qquad \Phi(x,s) = \begin{bmatrix} 0 \\ 0 \\ \phi_1(x) \\ 1 \end{bmatrix} \text{if} \quad s = 1 \tag{2.3}$$

Here, $\phi_0(x)$ and $\phi_1(x)$ represent two different feature representations extracted at two different scale windows - say for example, 50-pixel and 100-pixel tall people. Given training data triples (x_i, s_i, y_i) one could learn a single w that minimizes training error in (2.2) where $\Phi(x_i)$ is replaced by $\Phi(x_i, s_i)$.

It is straightforward to show that (2.2) reduces to *independent* SVM problems given the above multiresolution feature. It is equivalent to partitioning the dataset into small and large instances and training on each independently. This poses a problem since the detector scores for small and large detections need to be comparable. For example, one might expect that small-scale instances are harder to detect, and so such scores would generally be weaker than their large-scale counterparts. Comparable scores are essential to allow for proper non-max suppression between scales, contextual reasoning [17] and for ROC benchmark evaluation.

2.3.3 Multiscale multiresolution models

One mechanism of integrating two fixed-scale models is to also compute $\phi_0(x)$ for windows with $s = 1$. In other words, we can always resize a 100-pixel windows to 50-pixels and compute the resulting small-scale feature. This allows the large-resolution model to be *multiscale* in that features are computed multiple resolutions:

$$\Phi(x,s) = \begin{bmatrix} \phi_0(x) \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{if} \quad s = 0 \qquad \text{and} \qquad \Phi(x,s) = \begin{bmatrix} \phi_0(x) \\ 0 \\ \phi_1(x) \\ 1 \end{bmatrix} \text{if} \quad s = 1 \tag{2.4}$$

Note that because the coarse-scale features $\phi_0(x)$ are shared across both representations, the training problem no longer reduces to learning separate SVMs. In this case, distinct bias terms make scores for large and small instances comparable.

2.3.4 Multiresolution part models

One limitation of the above approach is that both small and large-scale models are encoded with a rigid template. Low-level descriptors such as HOG are invariant to small scale image

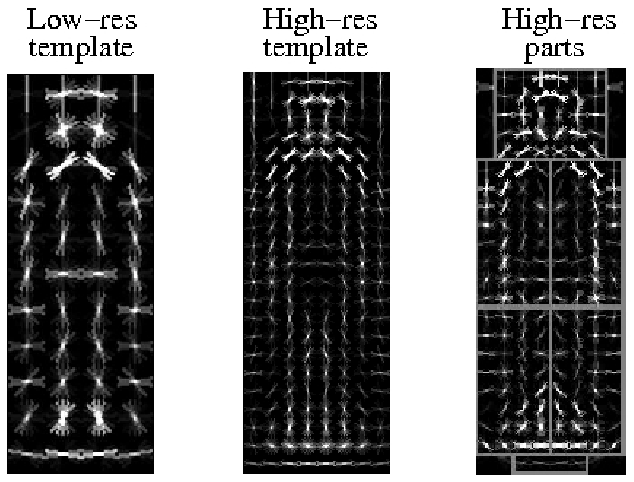


Figure 2.2: Finding large-scale instances. One might use a low-resolution template (shown on the **left**). Alternatively, to exploit the extra resolution of large-scale instances, one might define a high-resolution template (**middle**). Edges capturing the boundary of the body and head are blurred out due to variation in the postures of pedestrians in the training data. A more successful approach is to explicitly model the deformation with a part model (shown on the **right**), which learns sharper part templates.

deformation due to the local spatial binning of gradient values. However, this binning occurs at a fixed-size neighborhood (in our case, a neighborhood of 4×4 pixels). On the other hand, object deformations (such as the articulation of a pedestrian) occur at a scale relative to the *size of the instance.* This means that a HOG descriptor is likely invariant to the pose deformations of a 50-pixel pedestrian, but not a 100-pixel tall pedestrian.

To model pose variations at larger scales, we augment our large-scale model with a latent parameter capturing pose variation. Following the work of [33], we add a latent parameter z that specifies the location of a collection of parts. Given the z, we define $\phi_1(x, z)$ to be a vector of vectorized-HOG features extracted at the given part locations, appended with the part offsets themselves. This allows the corresponding parameters from w to encode part templates and part deformation parameters that penalize/favor certain part deformations over others.

$$\Phi(x,s,z) = \begin{bmatrix} \phi_0(x) \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{if} \quad s=0 \qquad \text{and} \qquad \Phi(x,s,z) = \begin{bmatrix} \phi_0(x) \\ 0 \\ \phi_1(x,z) \\ 1 \end{bmatrix} \text{if} \quad s=1 \tag{2.5}$$

The final classifier searches over latent values $f(x,s) = \max_z w \cdot \Phi(x,s,z)$:

$$f(x,s) = \begin{cases} w_0 \cdot \phi_0(x) + b_0 & \text{if} \quad s=0 \\ w_0 \cdot \phi_o(x) + \max_z w_1 \cdot \phi_1(x,z) + b_1 & \text{if} \quad s=1 \end{cases} \tag{2.6}$$

When scoring small instances, the above reduces to a standard linear template. When scoring large instances, the above requires a search over all part deformations, for the configuration that yields the maximum score. As in [33], we assume parts are independently positioned given a root location, equivalent to the standard "star" model assumptions in part-based models. This allows us to use dynamic programming to efficiently compute the max:

$$\max_z w_1 \cdot \phi_1(x,z) = \max_z \sum_j w_j \cdot \phi(x,z_j) + \sum_{j,k \in E} w_{jk} \cdot \phi(z_j,z_k) \tag{2.7}$$

where z_j is the location of part j, w_j is the template for part j, w_{jk} is a deformation model (spring) between part j and k, and E defines the edge structure in the star graph. We write $\phi(x,z_j)$ for the HOG feature extracted from location z_j and $\phi(z_j,z_k)$ for the squared relative offset between part j and k. Given training data triples (x_i,s_i,y_i), w can be trained with a latent SVM using the coordinate descent procedure outlined in [33] or the convex-concave procedure described in [104]. We use the publically-available coordinate descent code [1].

2.3.5 Latent multiresolution part models

One limitation of the above model is that training data is still specified in terms of a fixed, discrete size s_i - all instances are either 50 or 100 pixels tall. Given a training window of arbitrary height x_i, one might resize it to 50 or 100 pixels by quantization. The correct quantization may be ambiguous for datasets such as PASCAL where many images of people are truncated to head and shoulder shots [30] – here a small bounding box may be better described with a truncated, high-resolution model. When the training data x_i is given as set of bounding box coordinates, [33] shows that one can significantly improve performance by estimating a latent location and scale of a "hidden" bounding box that sufficiently overlaps the given ground-truth bounding box.

We augment this procedure to also estimate the "hidden resolution" s_i of a training instance x_i. Training examples that are large will not have any low-resolution (e.g., 50-pixel tall) bounding boxes that overlap the given ground-truth coordinates. In these cases, the resolution is fixed to $s_i = 1$ and is no longer latent. Similarly, training instances that are very small will not have any high-resolution bounding boxes with sufficient overlap. However, there will be a collection of training instances of "intermediate" size that could be processed as low or high-resolution instances. The values of s_i will be treated as latent and estimated through the latent SVM framework: starting with a random initialization of latent s_i and z_i values, (1) a model/weight-vector w is trained through convex optimization, and (2) the model is used to relabel an example x_i with a latent resolution state s_i and part location z_i that produces the best score.

Relationship to mixture models: It is relevant to compare our model to the mixture models described in [34]. One might view our multiresolution model as a mixture of two models. However, there are a number of important differences from [34]. Firstly, our components share many parameters, while those in [34] do not share any. For example, we use both low and high resolution instances to learn a low-res "root" template, while [34] only uses high-resolution instances. Secondly, the mixture component variable s_i is treated differently in our framework. At test time, this variable is *not* latent because we know the size of a putative window that is being scored. At train time, the variable is treated as latent for a subset of training instances whose resolution is ambiguous.

Extensions: Though we have described two-layer multi-resolution models, extensions to hierarchical models of three or more layers in straightforward. For example, the head part of a pedestrian may be composed of an eye, nose, and mouth parts. One would expect such a model to be even more accurate. Note that such a model is still efficient to score because the edge structure E is now a tree rather than a star model, which is still amenable to dynamic programming. Training a single resolution hierarchical part model poses a difficulty since it cannot exploit the many training and testing instances where the details, e.g., of the eyes and nose, are *not* resolvable. Our multiresolution formalism provides a framework to manage this complexity during both training and testing.

2.4 Multiresolution contextual models

We now augment our analysis of resolution to consider the effects of contextual reasoning. Our hypothesis, to be borne out by experiment, is that context plays a stronger role in detecting small-scale instances. Toward that end, we add a simple but effective contextual feature for pedestrian detection - ground plane estimation. Hoeim et. al. [43] clearly espouse the benefit of ground plane estimation for validating the observed locations and scales of putative detections. One approach would be to treat the ground plane as a latent variable to be estimated for each frame or video. We take a simpler approach and assume that the training and test data are collected in similar conditions, and so apply a ground-plane model learned from the training data at test time. We begin with the following assumptions:

1. The camera is aligned with the ground plane
2. Pedestrians have roughly the same height
3. Pedestrians are supported by a ground plane

Given the above and a standard perspective projection model, it is straightforward to show that there exists a linear relationship between the projected height of a detection (h) and the y-location of the lower edge of its bounding box in the image (y):

$$h = ay + b \tag{2.8}$$

Features: One reasonable contextual feature is to penalize the score of a detection in proportion to the squared deviation from the model:

$$(h - (ay + b))^2 = w_p \cdot \phi_p(x) \quad \text{where} \quad \phi_p(x) = \begin{bmatrix} h^2 & y^2 & hy & h & y & 1 \end{bmatrix}^T \tag{2.9}$$

where we have assumed the image features x include the location and height of the image window, and where model parameters w_p implicitly encode both the parameters of the ground plane and the amount to penalize detections which deviate from the ground plane model.

Our intuition says that low-resolution models should strongly penalize deviations because the local template will generate false positives due to its limited resolution. Alternately, the high-resolution model should not strongly penalize deviations because the local template is more accurate and the assumptions do not always hold (people are not all the same height). We investigate these possibilities experimentally using different encodings of our contextual features, including augmenting $\Phi(x, z, s)$ with a single set of perspective features $\phi_p(x)$ used across both low and high resolution models, or a separate set of features for each resolution ($\phi_p^0(x)$ and $\phi_p^1(x)$).

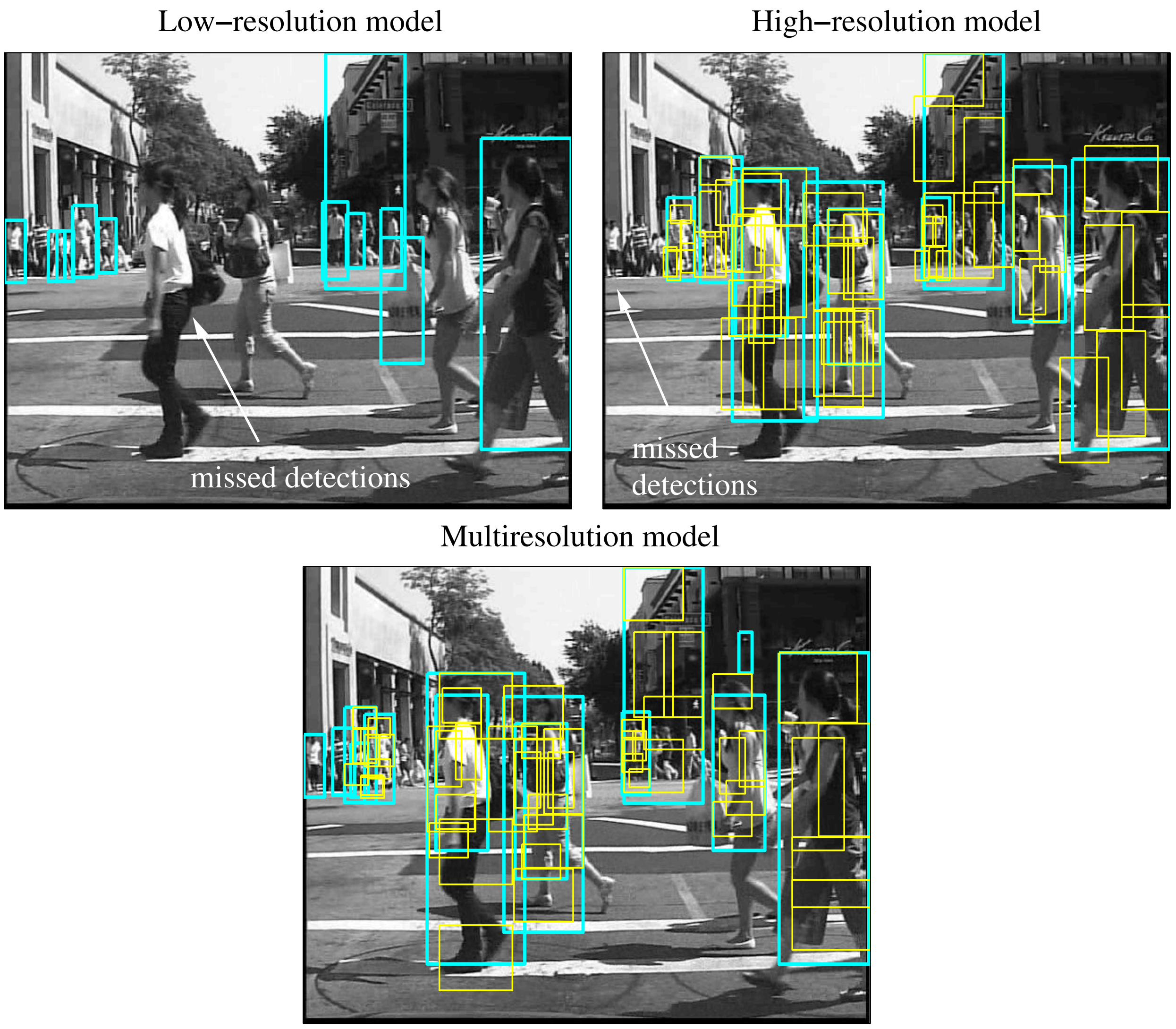


Figure 2.3: On the **left**, we show the result of our low-resolution rigid-template baseline. One can see it fails to detect large instances. On the **right**, we show detections of our high-resolution, part-based baseline, which fails to find small instances. On the **bottom**, we show detections of our multiresolution model that is able to detect both large and small instances. The threshold of each model is set to yield the same rate of FPPI of 1.2.

2.5 Experimental results

Implementation: We implemented our final context-augmented multiresolution model through fairly straightforward modification to the online multiscale part-based code [1]. In both the benchmark and diagnostic evaluation, we compare to the original code as a baseline. The contextual model decribed in the following results use scale-specific contextual features ($\phi_p^0(x)$ and $\phi_p^1(x)$), which we found slightly outperformed a single-scale contextual feature (though this is examined further in Sec.2.5.2).

2.5.1 Benchmark results

We submitted our system for evaluation on the Caltech Pedestrian Benchmark [22]. The benchmark curators scored our system using a battery of 11 experiments on a held-out testset, designed to analyze performance in different regimes depending on object scales, aspect ratios, and levels of occlusion (Fig. 2.4). The results are impressive - *our system outperforms all previously-reported methods*, across the entire range of FPPI (false positives per image) rates, in 10 out of 11 experiments. The sole experiment for which we do not win is the far-scale experiment, in which all detectors essentially fail. Even given our multiresolution model, finding extremely small objects is a fundamentally difficult problem because there is little information that can be extracted in such instances.

Our results are particularly impressive for the near-scale experiment, where we *halve* the previous-best miss rate at 1 FPPI [21]. Previous approaches, including the multiscale part-based model of [34], use fixed-resolution detectors that tend to be tuned for the small-scale regime so as to correctly fire on the set of small instances in this dataset. Our multiresolution model leverages the additional pixels available in large instances to significantly boost performance.

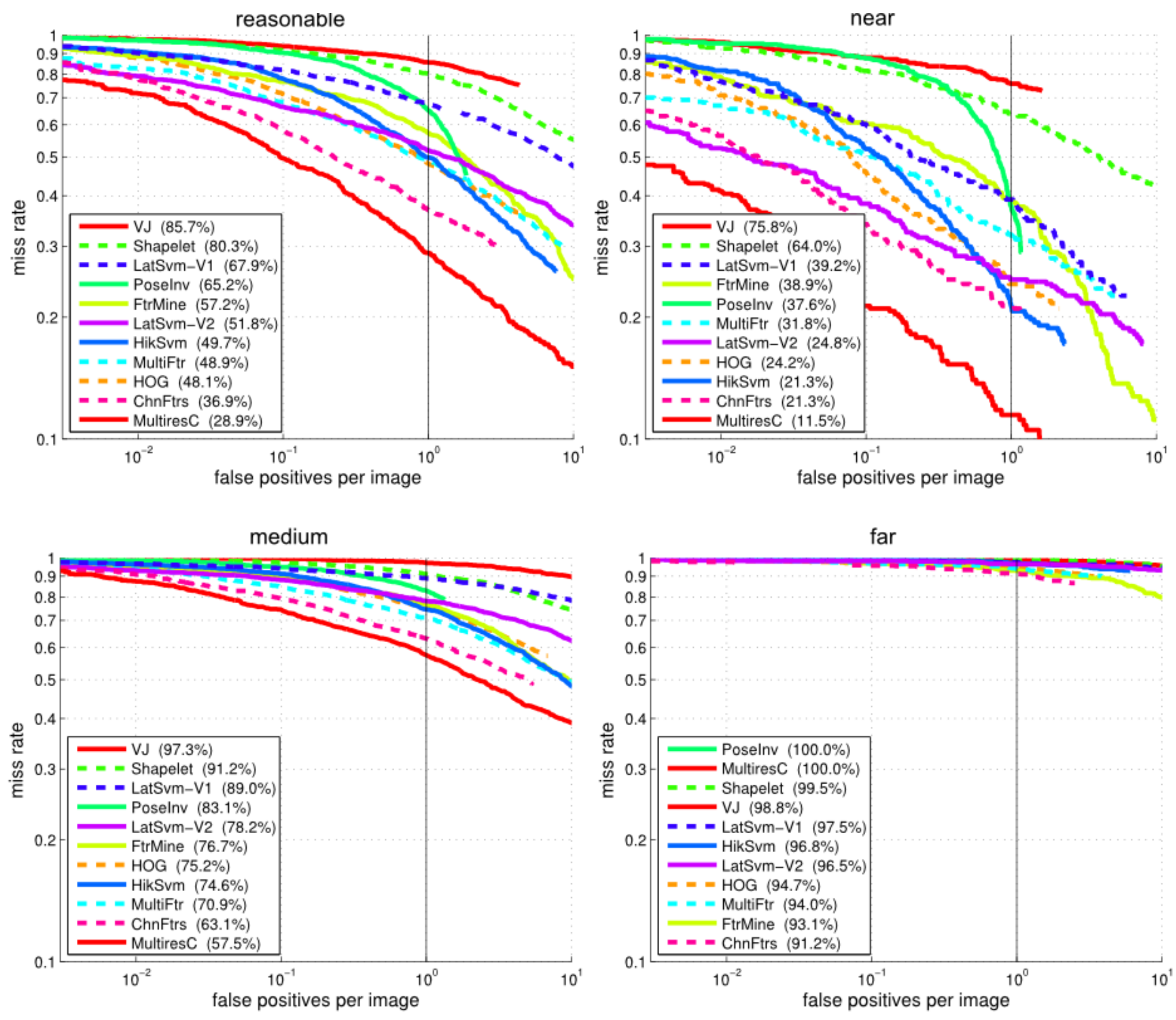


Figure 2.4: Benchmark results. From the upper left graph in **clockwise** direction, we show the results for *reasonable*, *near*, *far* and *medium* experiments, evaluated on test instances with various heights ($h > 30$, $h > 80$, $h < 30$, and $30 < h < 80$ and $h < 30$, respectively). Our context-augmented multiresolution model, labeled as **MultiresC**, significantly outperforms all previous systems in 10 out of the 11 benchmark experiments (all but the 'far' experiment').

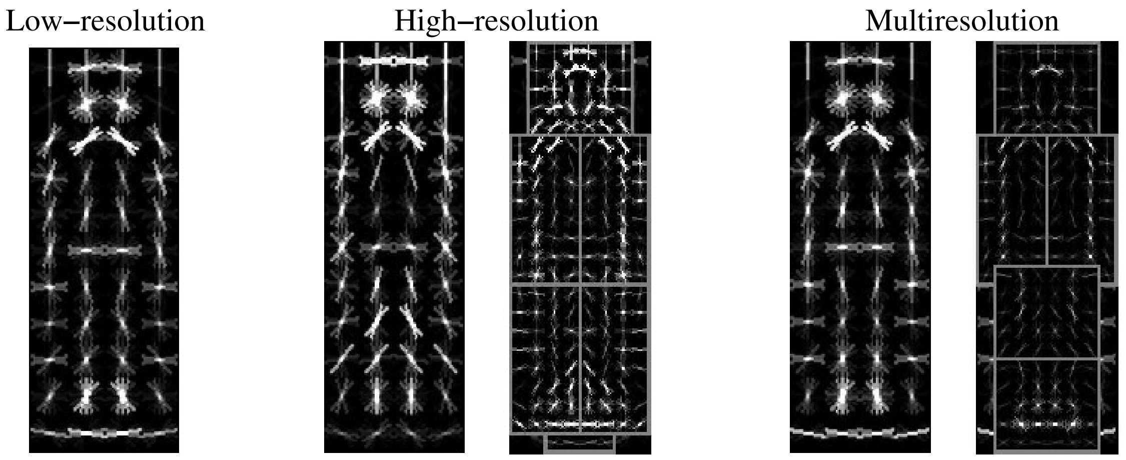


Figure 2.5: On the **left**, we visualize our low-resolution rigid-template. In the **middle**, we visualize the high-resolution part-based template of [33] trained on Caltech pedestrians. Note the root templates look different, as only a small portion of the training data (of high enough resolution) is used to train the part-model. On the **right**, we visualize the multiresolution model. Note that the root component looks similar to the low-resolution model. Also note that the parts overall have weaker weights. This suggests that much of the overall score of the multiresolution model is given by the root score. However, it is still able to detect both small and large instances as shown in our results.

2.5.2 Diagnostic experiments

To further analyze the performance of our system, we construct a set of diagnostic experiments by splitting up the publically-available Caltech Pedestrian training data into a disjoint set of training and validation videos. We defined this split pseudo-randomly, ensuring that similar numbers of people appeared in both sets. We compare to a high-resolution baseline (equivalent to the original part-based code [1]) and a low-resolution baseline (equivalent to a root-only model [13]), and a version of our multiresolution model without context. We visualize our baseline models in Fig. 2.5. All methods are trained and evaluated on the exact same data. To better interpret results, we threw out instances that were very small (< 30 pixels in height) or abnormal in aspect ratio (i.e. $h/w > 5$), as we view the latter as an artifact of annotating video by interpolation.

Overall: Overall, our multiresolution model outperforms baseline models. Our contextual model provides a small but noticeable improvement, reducing the missed detection rate from

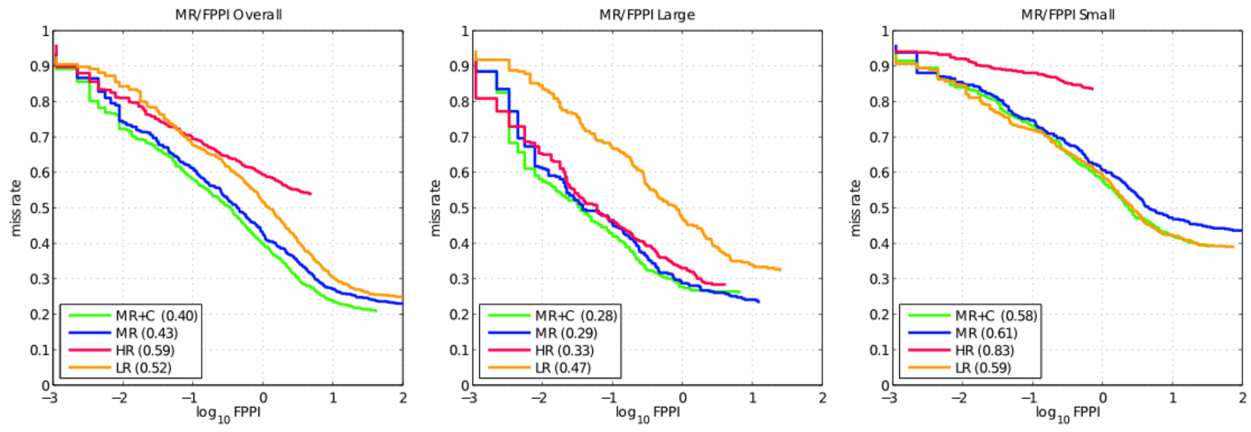


Figure 2.6: Results of diagnostic experiments. We compare results to fixed resolution baselines, where "LR" is a low-resolution rigid template and "HR" is a high-resolution part-based model. On the **left**, we show results evaluated on the full set of test instances from validation data. In the **middle**, we show results for large-instance (> 90 pixels). On the **right**, we show the results on small-instances (< 90 pixels). The "LR" template performs well on small instances, while the "HR" template performs well on large instances. Our multiresolution "MR" model exploits the best of both, in the appropriate regimes. Our context-augmented model "MR+C" provides a small improvement overall, but a noticable improvement when detecting small instances at a higher FPPI rate.

43% to 40%. We shall see that the majority of this improvement comes from detecting small-scale instances. Somewhat surprisingly, we see that a simple rigid template outperform a more sophisticated part model - 52% MD compared to 59%. One can attribute this to the fact that the part-based model has a fixed resolution of 88 pixels (selected through cross-validation), and so cannot detect any instances which are smaller. This significantly hurts performance as more than 80% of instances fall in this small category. However, one may suspect that the part-model should perform better when evaluating results on test instances that are 88 pixels or taller.

Detecting large instances: When evaluating on large instances (> 90 pixels in height), our multiresolution model performs similarly to the high-resolution part-based model. Both of these models provide a stark improvement over a low-resolution rigid template. We also see that perspective context provides no observable improvement. One might argue that this is due to a weak contextual feature, but we next show that it does provide a strong

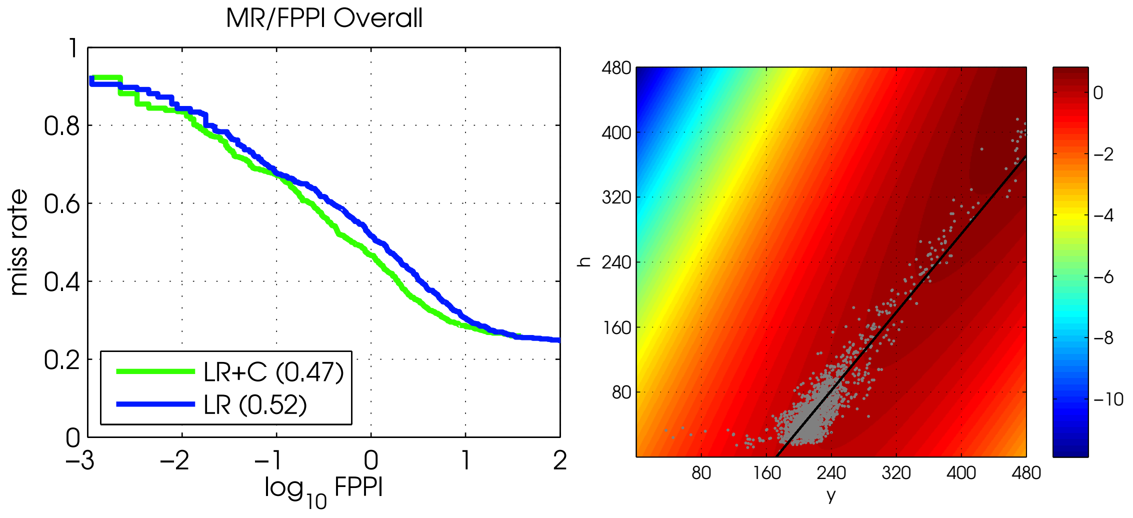


Figure 2.7: We show the effectiveness of our perspective features on low-resolution models. Overall performance increases from 51% MD to 46% MD. We visualize our perspective features on the **right**. We plot the distribution of h and y (bounding box height and image-y locations) in the ground truth data, and plot the score $w_p \cdot \phi_p(x)$ as a function h and y. We also display the distribution of ground truth (visualized with a point cloud) along with its linear fit. We see that the learned contextual features penalize detections whose heights and image-y locations are not consistent with the ground plane.

improvement for small scale detections.

Detecting small instances: When evaluating on small instances ($<$ 90 pixels in height), we see that the part-based model performs quite poorly, as it is unable to detect the majority of test instances which are small. Our multiresolution model performs slightly worse than a low-resolution model (61% compared to 59%). Perspective features provide a noticeable improvement for our multiresolution model, increasing performance from 61% MD to 58%.

Context features: To verify that our contextual features are indeed reasonable, we analyze the benefit of our contextual features on a low-resolution model. We see a noticeable reduction in the MD rate from 51% to 46%, suggesting our contextual features are indeed fairly effective. Their effect is diminished in our multiresolution model because the part-based model is able to better score large-scale instances, reducing the need for score adjustment using context.

2.6 Conclusion

We describe a simple but effective framework for merging different object representations, tuned for different scale-regimes, into a single coherent multi-resolution model. Our model exploits the intuition that large instances should be easier to score, implying that one should adapt representations at the instance-level. We also demonstrate that context should be similarly adapted at the instance-level. Smaller objects are more difficult to recognize, and it is under this regime that one should expect to see the largest gains from contextual reasoning. We demonstrate impressive results on the difficult but practical problem of finding large and small pedestrians from a moving vehicle.

Chapter 3

Exploring weak stabilization for motion feature extraction

3.1 Introduction

The previous chapter dicussed multiresolution methods for finding small people. In this chapter, we will explore an alternate cue for finding small instances of people - visual motion.

Most approaches for visual recognition focus on the static-image setting; indeed, a common method for detecting objects in video is to run an image-based detector on each frame. Significant progress has been made in static-image object detection over the past few years, in large part due to the improvement of low-level features coupled with classifiers such as SVMs [14] and boosting [89]. In this chapter, we explore the motion counterpart for pedestrian detection in video. We show that one can exploit simple motion features to significantly increase detection accuracy with little additional computation.

Image motion observed in videos is the result of several sources, Figure 3.1. We classify

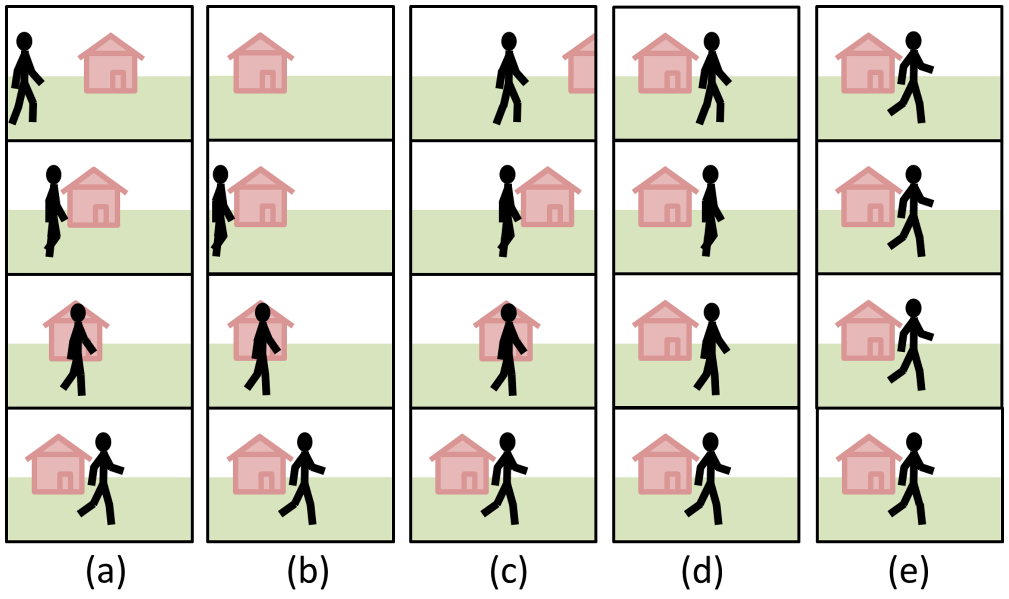


Figure 3.1: Illustration of various types of video stabilization: (a) no stabilization, (b) camera motion stabilization, (c) object-centric motion stabilization, (d) camera and object-centric motion stabilization, and (e) full stabilization of camera, object-centric, and part-centric motion. We posit that for detecting articulated objects such as people the majority of useful motion information is contained in part-centric motion. We therefore attempt to stabilize both camera and object-centric motion, as in (d).

image motion into three types using a stationary world coordinate frame and a moving object coordinate frame. Camera-centric motion is the movement of the camera with respect to the world. Object-centric motion is the movement of the object centroid with respect to the world. Finally, part-centric motion is the movement of object parts with respect to the object. These three types of motion provide different cues for recognition.

Prior work makes different assumptions about which motion types are useful versus nuisance factors. A simple approach is to directly compute image motion features on raw video. In this case, the observed image motion contains camera-, object-, and part-centric motion. Methods that define motion features using optical flow or spacetime gradients often take this route [92]. One can partly remove camera motion by looking at differences of flow [15]. A more direct approach is to simply compute motion features on a stationary camera, such as [90]. Such motion features encode both object- and part-centric motion. The large body of techniques that rely on background subtraction take this approach [71]. When the camera is moving, one may try to register frames using a homography or egomotion estimation [42, 46], which removes some camera-centric motion but can be challenging for dynamic scenes or those with complex 3D geometry. Finally, other techniques compute optical flow in an object-centric coordinate frame [25]; Figure 3.1(c) shows that such an approach actually encodes both camera- and part-centric motion.

In this chapter, we posit (and verify by experiment) that the majority of useful motion information for detecting articulated objects such as people is contained in part-centric motion. As shown in Figure 3.1, there are numerous types of video stabilization. To allow the temporal features to easily extract part-centric motion information, we attempt to stabilize both camera and object-centric motion, Figure 3.1(d). We accomplish this by using *coarse-scale* optical flow to align a sequence of image frames. Weak stabilization using coarse-scale flow has the benefit of aligning large objects such as the background or a person's body without removing detailed motion such as an object's parts, Figure 3.1(d,e). While artifacts may exists around large flow discontinuities, we demonstrate that coarse-scale flow is robust in practice.

We use temporal difference features to capture the part-centric motion that remains after weak stabilization. While features based on fine-scale optical flow [25, 15] may be extracted from the stabilized frames, fine-scale flow is notoriously difficult to extract for small parts such as arms [9]. We demonstrate that when sampled at the proper temporal intervals, simple temporal difference features are an effective alternative capable of achieving state-of-the-art results.

We perform a thorough evaluation of motion features for people detection in video. We focus on detecting pedestrians in moving cameras [23] as well as pose estimation from static cameras [2]. We demonstrate significant improvements from integrating our motion features into three distinct approaches: rigid SVM detectors defined on HOG features [14], articulated part models defined on HOG features [34, 101], and boosted detectors defined on channel features [21]. Notably, we report a five-fold reduction in false positives at fixed detection rates on the Caltech Pedestrian Benchmark, a significant improvement over prior art.

Finally, we perform an exhaustive sweep over various parameter settings of our model to analyze what aspects are important. We find it crucial to (1) compute optical-flow at the right level of coarseness to provide camera and object-centric stabilization, (2) compute difference features over long time scales (because motion over pairs of successive frames may be too subtle to measure) and (3) normalize motion features appropriately for use with linear SVMs.

3.2 Related Work

Optical-flow-based features: A popular strategy for video-based recognition is to extend static image features into the temporal domain through use of optical flow. Examples include spatially blurred flow fields [25] or histograms of optical flow vectors [15, 91]. In particular, Dalal *et al.* explored flow-augmented versions of their HOG descriptor [15] termed histograms of flow (HOF). Although HOF performed well for classification, Dalal's thesis admitted that it under-performed a HOG template when evaluated for detection [12]. Walk *et al.* [91] proposed a number of modifications to the HOF features that resulted in modest gains in detection performance. Recognizing the difficulty of accurate fine-scale flow estimation (aperture problem, singularities, etc.), [80] proposed directly comparing motion fields without explicit computation of flows.

Temporal-difference features: Temporal differencing, or temporal gradient features, date back to the early work of [3]. Since two-frame differencing might be too weak to produce a signal for slow-moving objects, [11, 48] describe approaches for multi-frame differencing. A related and popular approach is histograms of spacetime gradients [105, 51]. For stationary cameras, temporal difference features can be computed on background models, yielding background-subtraction masks [71]. Our approach can be seen as a combination of optical-flow and temporal differencing as we compute differences on spacetime windows that are weakly-stabilized with coarse optical flow.

Action classification: Many of the above motion features have been explored in the context of action classification [20, 51]. In particular, [92] performs a thorough evaluation of motion descriptors, discovering that histograms of flow perform well. For our setting of detecting low resolution objects in videos, traditional flow fails because small movements are difficult to estimate reliably. While effective for behavior classification, space-time interest points have not proven useful for object detection.

Tracking: An alternate use of temporal information to improve detection reliability is to explicitly track objects. For example, detection may be improved by tracking repeated detections [4]. Most trackers tend to define motion models on static image features, although exceptions do exist [31]. Impressive results have also been shown on a system wide integration of detectors [28, 95, 41]. Such approaches are orthogonal to ours as we aim to improve the quality of the detections themselves through use of more informative image features.

3.3 Approach

In this section, we describe our basic approach to motion feature extraction. We begin by discussing basic notation and static features. We then describe our approach to weakly-stabilizing video frames and our resulting motion features. Results are provided in the following section.

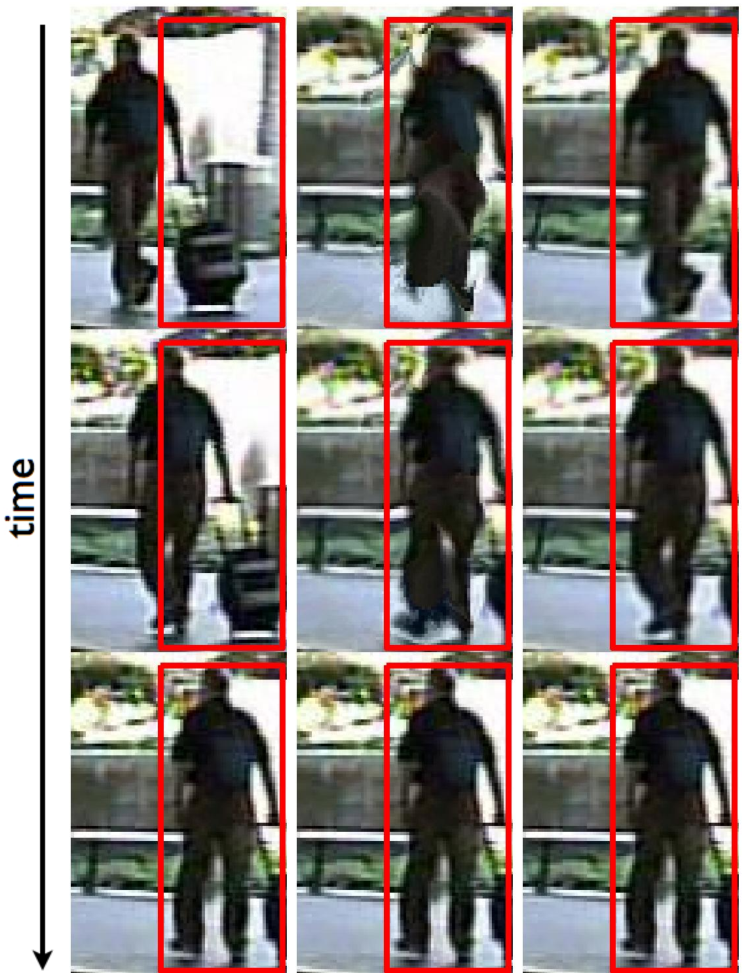
Notation: Let $\mathcal{I}_t$ denotes the t-th frame of a given video and I_t denote an image patch from $\mathcal{I}_t$. The spatial extent of I_t in the frame is implicitly defined by the detection task. For pedestrian detection, I_t is a fixed-size 32×64 pixel patch. To detect people at different scales we use an efficiently computed image pyramid [19].

Static features: In addition to the motion features introduced below, we use one of two sets of static features densely computed on the *current* frame. Our first set of static features are the *channel features* described in [21]. As in [21], our channels include color (3 channels), gradient magnitude (1 channel) and gradient quantized by orientation (6 channels). Our second type of static features is the commonly used Histogram of Oriented Gradients (HOG) descriptor [14]. Specifically, we compute histograms of gradients using 9 orientations on an 8×16 grid of 4×4 cells.

3.3.1 Stabilizing videos

Our goal is to compute motion features based on part-centric motion, such as the movement of a person's limbs. This requires weakly stabilizing image frames to remove both camera and object-centric motion while preserving the part-centric motion. We accomplish this by using *coarse-scale* optical flow to align a sequence of frames.

We estimate optical flow using the approach of Lucas-Kanade [57] but applied in a somewhat non-standard manner. Lucas-Kanade proposed a differential approach to flow estimation that is commonly implemented hierarchically. A window radius σ controls the scale of the flow. Typically, σ must be large enough to provide reliable local flow estimate but small enough to capture fine motions. Instead, *coarse* flow can be computed using a large radius σ. This offers dual advantages: the flow estimates are both more reliable and faster to compute.



(a) Raw (b) =4 (c) =32

Figure 3.2: Stabilization using coarse-scale LK flows. We show temporally distant 3-frame sequences stabilized onto the last frame (bottom row). The red box in each frame is the location of the person in the last frame. (**a**) In the raw video, the person shifts from left to right due to camera and object motion. (**b**) Using fine-scale LK flows, the overall body is stabilized onto the last frame at the cost of distortion in body parts (most visible at the heads and legs of the top row). (**c**) Using coarse-scale LK flows the warped images are aligned in terms of the overall body location while still preserving clear motions of body parts.

We compute Lucas-Kanade flows with σ typically ranging from 8 to 32 pixels (16×16 to 64×64 windows). We denote the computed flow field from frame $\mathcal{I}_t$ to frame $\mathcal{I}_{t-1}$ as $W_{t,t-1}$. $\mathcal{I}_{t-1,t}$ is frame $\mathcal{I}_{t-1}$ warped to frame $\mathcal{I}_t$ using the flow field $W_{t,t-1}$. We write an image patch from the warped image as $I_{t-1,t}$. In practice, we find $W_{t,t-1}$ stabilizes the majority of motion due to camera and object-centric motion, as shown in Figure 3.2. Computing the coarse flows is fast (no need to compute flow at finest scale) and fairly robust (due to the large σ).

When stabilizing across multiple frames, we compute the global motion $W_{t,t-n}$ by progressively warping and summing pairwise flow fields. We found this to work better in practice than computing the potentially large flow directly between frames $\mathcal{I}_t$ and $\mathcal{I}_{t-n}$.

3.3.2 Motion features

Given (weakly) stabilized image frames, we propose the use of simple temporal differencing or temporal gradient features. We now describe the numerous variants that we experimentally evaluate. The temporal gradient is defined as the difference between two frames,

$$D^{\sigma} = I_t - I_{t-1,t}, \tag{3.1}$$

where σ is the scale of the computed flow. Because σ is tuned to be roughly the size of an object, we expect the temporal gradient to contain useful cues about nonrigid object motion that are helpful for detection, as in Figure 3.3. We denote temporal gradient on unstabilized frames as D^{US}:

$$D^{US} = I_t - I_{t-1} \tag{3.2}$$

Using multiple frames: We previously defined the difference features over pairs of frames. In many instances, the amount of motion observed between subsequent frames may be quite small, especially with slow moving objects. Consider Figure 3.2; it is hard to see the difference

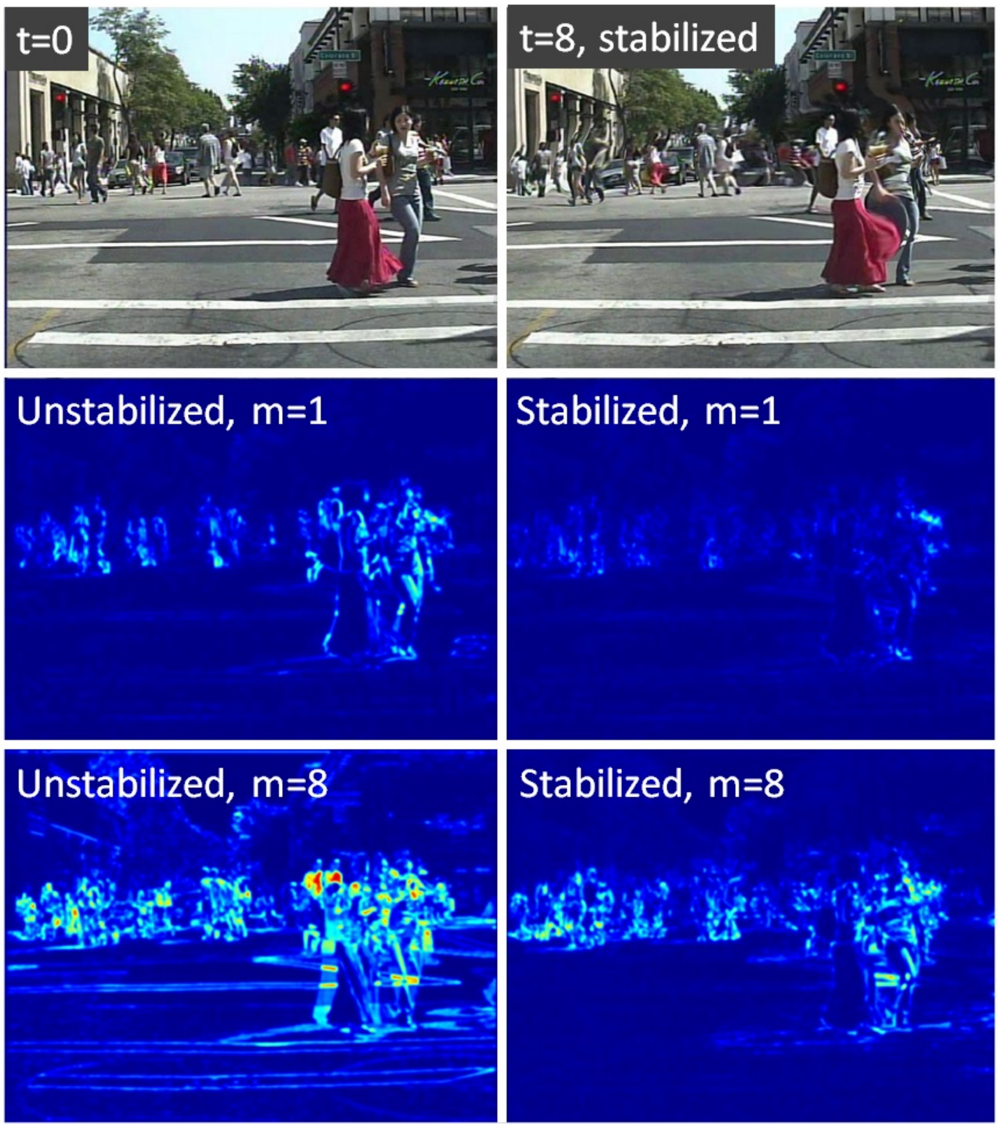


Figure 3.3: Example temporal frame differences using unstabilized and weakly stabilized frames spaced one frame apart ($m = 1$) and 8 frames apart ($m = 8$). When $m = 1$ there exists minimal temporal information. With larger frame spans ($m = 8$) temporal differences appear. However, weak stabilization is needed to remove non-informative differences resulting from camera and object motion.

in poses between temporally adjacent frames. We alleviate this by considering multiple frames, or frames spaced further apart. Next, we define a family of multi-frame approaches.

First, we consider the simple approach of computing multiple frame differences between the current frame and $k = n/m$ other frames spaced apart temporally by m frames from $t - m$ to $t - n$. We refer to m as the *frame skip* and n as the *frame span*.

$$D_0^\sigma(n,m) = \begin{bmatrix} I_t - I_{t-1m,t} \\ I_t - I_{t-2m,t} \\ \vdots \\ I_t - I_{t-km,t} \end{bmatrix} \tag{3.3}$$

Using this notation, D^σ in Equation (3.1) computed from only two neighboring frames is equivalent to $D_0^\sigma(1,1)$.

Another approach is to compute the set of differences between neighboring frames within a multiframe set,

$$D_1^\sigma(n,m) = \begin{bmatrix} I_t - I_{t-m,t} \\ I_{t-m,t} - I_{t-2m,t} \\ \vdots \\ I_{t-(n-m),t} - I_{t-n,t} \end{bmatrix} \tag{3.4}$$

Finally, we may also compute the difference between the mean frame M_t and the neighboring frames,

$$D_M^\sigma(n,m) = \begin{bmatrix} M_t - I_{t-0m,t} \\ M_t - I_{t-1m,t} \\ \vdots \\ M_t - I_{t-km,t} \end{bmatrix}, \tag{3.5}$$

where $M_t = \frac{1}{k+1} \sum_{i=0}^{k} I_{t-im,t}$

Rectified features: Previously, we defined our temporal difference features using the signed temporal gradient. Several other possibilities also exist for encoding the temporal differences, such as using the absolute value of the temporal gradient or using rectified gradients. Rectified gradients compute two features for each pixel's temporal gradient dt corresponding to $\max(0, dt)$ and $\max(0, -dt)$. The motivation for this is that the sign of the gradient might provide additional information for detection (e.g. people often have darker hair color or clothing than the background).

Feature pooling: To add a small amount of spatial invariance, all of our features are pooled over a $c \times c$ sized rectangular window. In all our experiments our pooling size is 4×4. The pooling is the same as for the static features.

Feature normalization: The contrast between a person to be detected and the background may vary due to lighting, background texture or clothing. This affects both static and temporal difference features. Static features such as HOG [14] account for this using feature normalization. We follow a similar approach, but extended to spatio-temporal blocks. After pooling our difference features over $c \times c$ neighborhoods, we construct overlapping $s \times s \times t$ blocks of cells with spatial extent $s = 2$ and temporal extent $t = 2$ (analogous to R-HOG, but extended in time). We then $L1$ normalize each block feature (which we found to outperform $L2$ normalization). To improve performance, we found it important to clip the computed $L1$ norm of each block to have a maximum value of .05. Finally, following the approach of [34], we use the average of eight normalized values (computed from overlapping spacetime blocks) as the final feature.

3.4 Experimental results

In this section, we present a thorough evaluation of the family of features described above. We evaluate our results on two datasets, the Caltech Pedestrian dataset [23] and the MindsEye

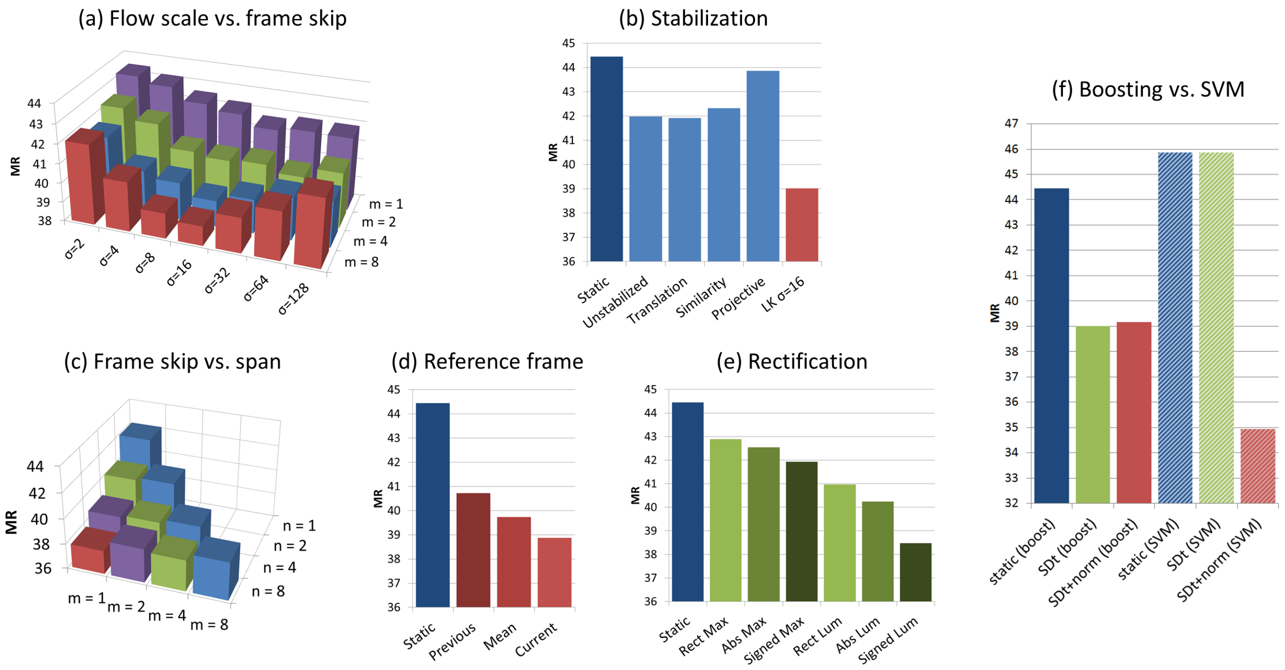


Figure 3.4: Results for various parameter sweeps on the Caltech pedestrian dataset. These include (a) adjusting the flow scale σ vs. the frame skip m, (b) other forms of stabilization, (c) frame skip m vs. frame span n, (d) various types of reference frames for computing $D(m, n)$, (e) different types of rectification for utilizing the color channels, and (f) boosting vs. SVM results with and without normalization. The best results, $D_0^{16}(8, 4)$, are achieved using $\sigma = 16$, $m = 4$, $n = 8$, the current frame as reference, and the signed temporal differences of the luminance channel. The SVM classifier outperforms the boosting classifier when normalization is used. Normalization has no effect on the boosting classifier.

dataset [2]. We begin by exploring the feature parameter space on the task of pedestrian detection using a boosting classifier [21]. For the use of linear SVM [14] classifiers we show that normalizing features is crucial. With the optimal setting, state-of-the-art results are shown using boosting and linear classifiers. We conclude our experimental results by showing promising results on the challenging task of part detection using the MindsEye dataset [2].

3.4.1 Pedestrian detection

In this section, we explore various parameter settings on the Caltech Pedestrian dataset [23], which consists of 10 hours of real-world video footage from a car-mounted camera. The full dataset contains over 350,000 pedestrian detections. As is recommended practice [23], we train and evaluate using every 30^{th} frame and a smaller "reasonable" subset of bounding boxes containing pedestrians 50 pixels or taller and with limited occlusion. For boosted classifiers, we average results over 20 trials with different random seeds to increase their statistical significance.

We measure accuracy using the standard log-average miss rate for the detections [23], which is computed by averaging the miss rate at nine false positives per image (FPPI) rates evenly spaced between 10^{-2} to 10^{0}. A detection is labeled as correct if the area of overlap is greater than 50%.

We implement several baselines. The result of Dollár *et al.* [21], as reported in [23], is a 56% log-average miss rate using only static features and trained on the INRIA dataset [14]. Retraining on the Caltech training set reduced this error to 51%, which is close to the best reported results. By shrinking the model size from 128×64 to 64×32 and excluding occluded pedestrians from the training set we were able to reduce this rate to 45%. Likewise, using code from [34] we trained a HOG-SVM detector [14]. Again excluding occluded pedestrians, using a reduced model size, and shrinking the default HOG cell size to 4×4 pixels, we achieve 46% miss rate. *Our baselines slightly outperform the best reported results on the Caltech dataset.*

We now describe experiments testing each parameter. We perform our sweeps using boosting and the 10 static channel features described in Section 3.3. We explore each parameter sequentially while holding the others constant. For reference, we also always show the performance of our static detector. Lastly, we combine the optimal temporal features found for boosting with the static HOG features for use by linear SVMs. For the sweeps in Fig. 3.4 we used a slightly simplified evaluation criterion, resulting in minor differences from our final numbers reported in Fig. 3.5 (generated using the official evaluation code available from [23]).

Optical flow scale vs. frame skip: We first explore the space of two parameters; the scale of LK flows, σ, and the *skip* between two frames used to compute the temporal difference, m, see Fig. 3.4(a). For these experiments, we only use two frames, with the span n equal to m. We use D_0, where the first frame is the reference frame when computing differences. Observe that there exists a coherent relationship between miss rates and these two parameters. When the pair of frames are temporally nearby, stabilization plays a smaller role, since objects are relatively well aligned even without stabilization. As we increase the skip m between the pair of frames, stabilization becomes critical. We fix $\sigma = 16$ for all remaining experiments.

Ideally, the optical flow scale should roughly cover an object, and so would be defined relative to the size of the candidate window being evaluated. For simplicity, we implemented a fixed scale in our experiments, which still worked well because our datasets tend to contain objects at a single scale. Moreover, Fig. 3.4(a) shows stable performance over two octaves in scale space, indicating that precise scale selection may not be necessary in general.

Other forms of stabilization: In addition, we explored global 2D transformations for stabilizing videos including translation, similarity, and projective transformations. Our stabilization outperforms these considerably, see Fig. 3.4(b).

Multiframe: Given a fixed scale $\sigma = 16$, we now examine the question of the optimal multiframe span n, skip m, and reference frame. Certain combinations are not possible ($m > n$) and so cannot be evaluated. We find that a large span $n = 8$ and small skip value $m = 1$ performs best, although a larger skip $m = 4$ also does well, see Fig. 3.4(c). Given the reduction in computational complexity of $D(8, 4)$ over $D(8, 1)$, we fix $n = 8$ and $m = 4$. Using these settings, we find using the current frame, $\mathcal{I}_t$, as the reference achieves the best result, see Fig. 3.4(d). This yields the final multiframe motion feature of $D_0(n = 8, m = 4)$.

Rectification: We examine various strategies for feature rectification in Fig. 3.4(e), using three temporal differences across the LUV color channels. The "Max" scheme uses the maximum temporal difference across the 3 channels, while the "Lum" scheme just uses the luminance (L) channel. "Rect" refers to rectified features that are created by appending the absolute value of the positive and negative components of the difference feature $D_0(8, 4)$. "Abs" refers to simply taking the absolute value of the difference feature, while "Signed" refers to keeping the original signed feature. We see in Fig. 3.4(e) that the signed luminance feature outperforms all the other variants.

Normalization: We evaluate the impact of feature normalization in Fig. 3.4(f). The normalization has minimal effect on the performance of the boosting classifier, presumably because boosting classifiers can train more flexible decision boundaries that perform implicit normalization. However, explicit normalization appears vital for linear SVMs. Similar finding have been shown for static features such as HOG [14].

Previous work: In Fig. 3.5 we compare with previous work including 'MultiFtr+Motion' [91] (which uses motion features) and 'MultiresC' [68] (which uses static features trained on the same data as [23]). Our models considerably outperform prior work, achieving a five-fold reduction in false positives. Both boosting and SVM classifiers perform well, each being optimal for different ranges of FPPI. Fig. 3.6 shows several examples of detections using our approach compared to using static features alone. Several false detections are removed

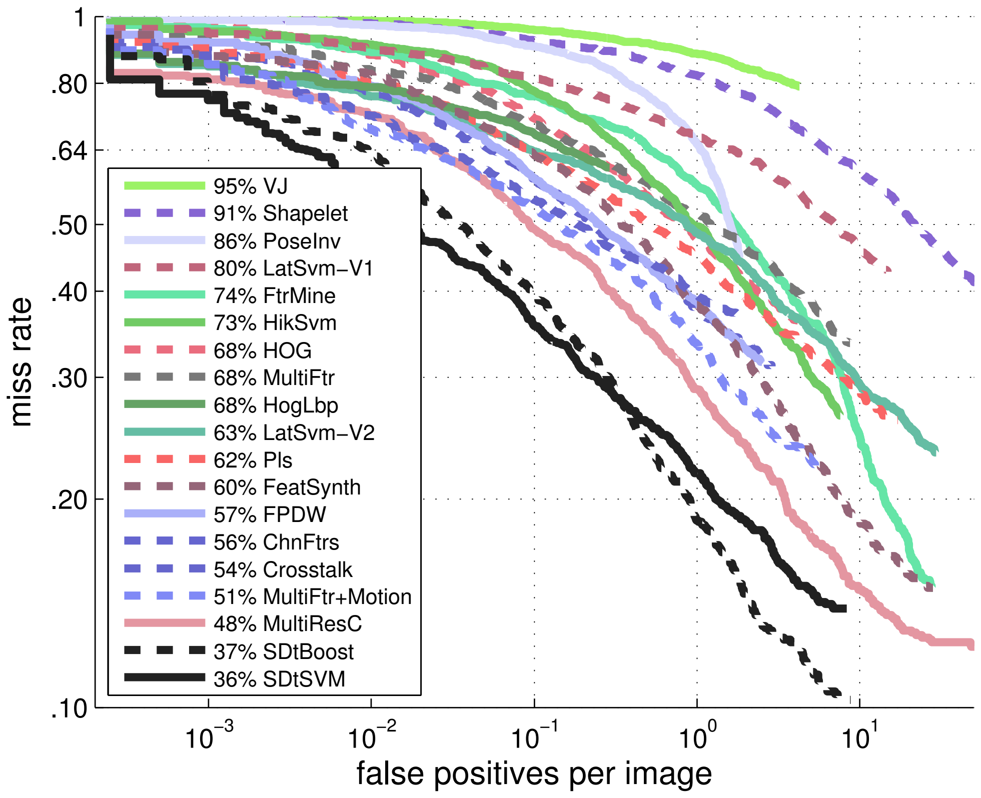


Figure 3.5: Comparison of log-average miss rate vs. False Positives Per Image (FPPI) between our approaches and previous methods on Caltech [23]. Our new temporal features lead to a significant improvement across all FPPI rates.

around the car's boundary as temporal features remove the ambiguities. Temporal features can also help discover missed detections, such as the pedestrian riding a bicycle in the second row.

3.4.2 Part detection

The MindsEye video dataset [2] is a large collection containing hundreds of hours of video capturing everyday outdoor human interactions for military surveillance scenarios. It is one of the largest available datasets for multi-person pose estimation and multi-person action recognition (Fig. 3.7). Though scripted, it is a challenging testbed for video analysis. We have annotated human poses in a collection of 7 video clips with each 30-100 seconds in duration. The annotated frames are evenly split into training and testing, and used to evaluate the ability of our motion features to perform human pose estimation in video sequences.

Baseline articulated part model: We describe our baseline articulated part model [101], and show how to extend it to incorporate our motion features. Let $l_i = (x_i, y_i)$ be the pixel location of part i. Given an image I, we score a collection of part locations $l = \{l_i\}$

$$\text{score}(I, l) = \sum_i w_i \cdot \phi(I, l_i) + w_s \cdot \text{spatial}(l) \tag{3.6}$$

where $\phi(I, l_i)$ is a HOG descriptor extracted from pixel location l_i in image I. The first term in (3.6) is an appearance model that computes the local score of placing filter w_i at location l_i using an inner-product. The second term is a shape prior that favors particular spatial arrangements of parts over others. From our perspective, we can be agnostic to its form so long as it is linearly parametrized and there exist tractable algorithms for computing the best scoring configuration $\max \text{score}_l(I, l)$. [101] describes efficient dynamic programming algorithms for inference, as well as efficient quadratic programming solvers for learning parameters $\{w_i, w_s\}$ given labeled training data.

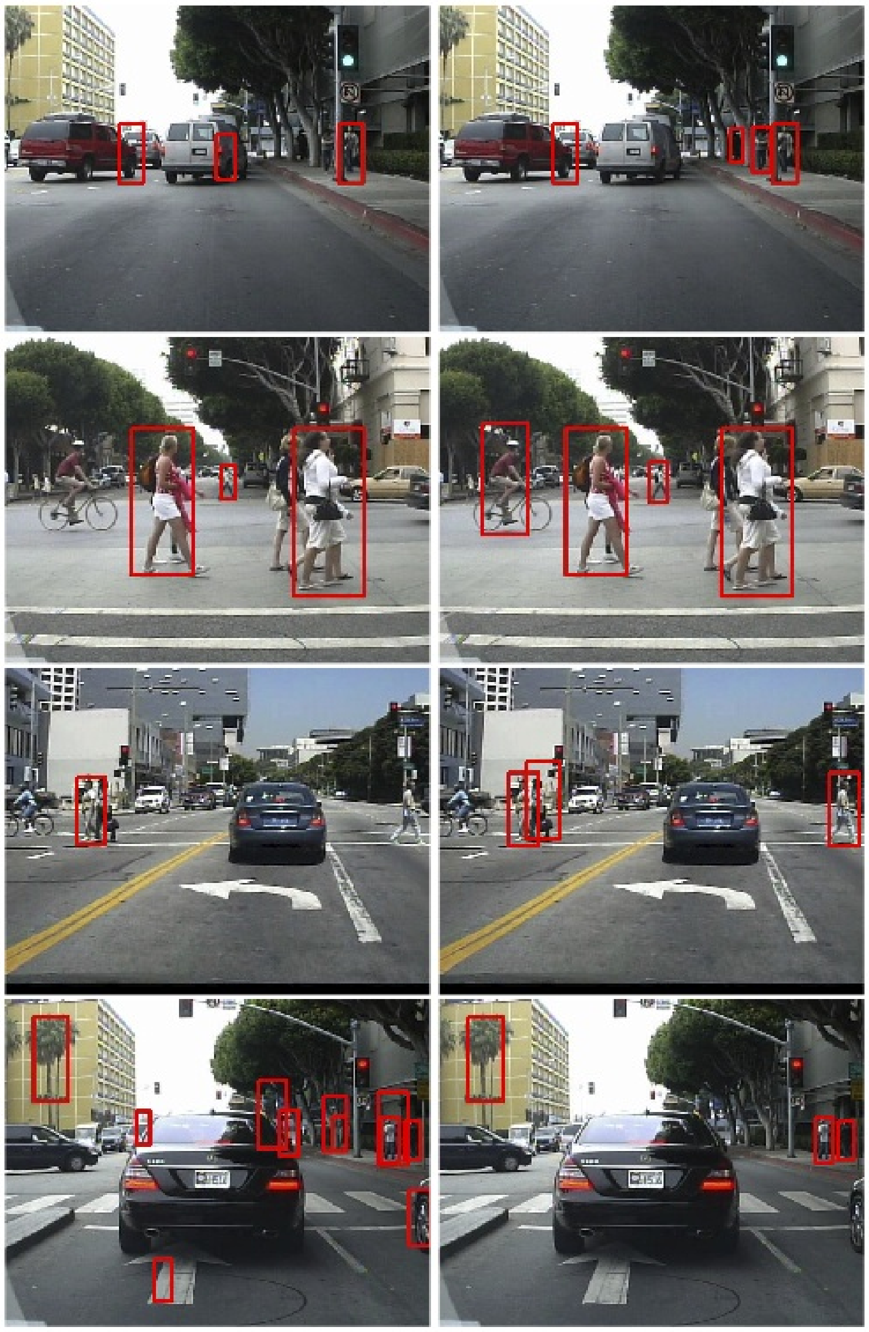


Figure 3.6: In the each row, we compare the results of two models; one trained only with static features (**left**), and the other trained with both static and our motion features (**right**). Note that our motion features help detect instances that are considered hard due to abnormal pose (biking) or occlusion, and significantly reduce false positives.

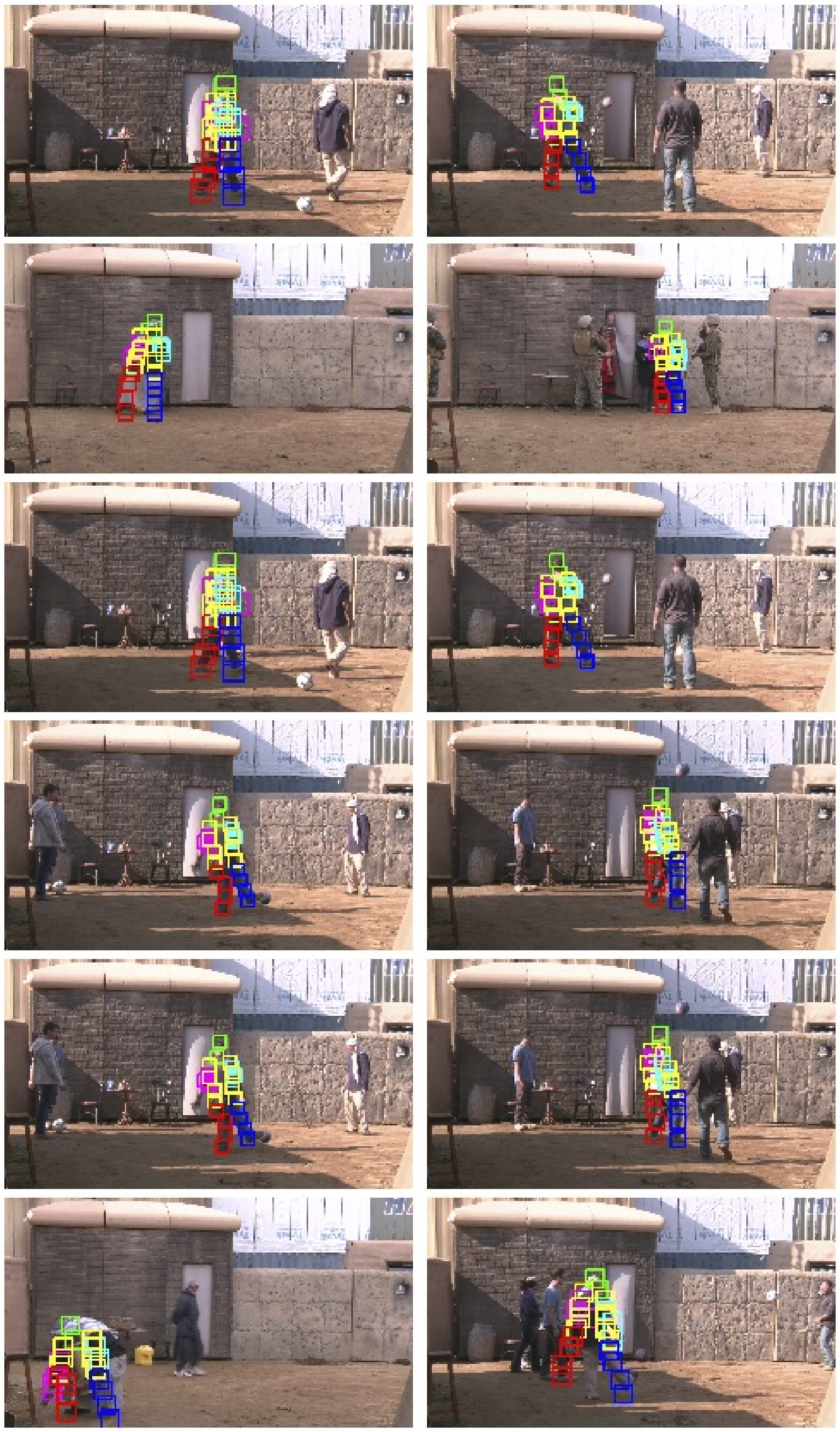


Figure 3.7: Pose estimation on MindsEye test images. We show estimates from the pose model of [101] trained using our motion features. It outperforms static features, especially for instances with large motion, e.g. playing with a ball. The last row shows failure cases.

Features	HOG	HOG+Motion
Head	71.50%	**76.00%**
Upper arms	65.00%	**68.25%**
Lower arms	35.25%	**39.25%**
Upper legs	62.50%	**65.50%**
Lower legs	60.75%	**61.75%**
Overall	57.07%	**59.93%**

Table 3.1: Augmenting an articulated part model with our motion features produces consistently better part localizations. The gain from static features are not as dramatic as the result on Caltech, since other challenges, such as self-occlusion, inter-person occlusion, and a wider variety of poses, plays a role. Each body part (e.g., upper arm) contains 2 keypoints and is evaluated using standard criteria [101]. "Overall" refers to the average across all keypoints, making sure there is no double-counting.

Motion features: For our experiments, we simply augment the appearance descriptor to include both HOG and our motion feature:

$$\phi(I, l_i) = \begin{bmatrix} HOG[I, l_i] \\ D_0(8,4)[I, l_i] \end{bmatrix} \tag{3.7}$$

The above formulation allows us to easily incorporate our motion features into the existing pipeline at both test-time and train-time. Since the people in the MindsEye dataset are significantly larger, we increased σ to 50.

Evaluation: We augmented the publicly-available code of [101] to use our motion features. We trained both a static-image pose detector and motion-augmented pose detector using the exact same training data, and present results in Fig. 3.7 and Table 3.1. For upper body parts, we see a large improvement in part localization accuracy (as measured by the fraction of times a predicted joint sufficiently overlaps the ground-truth). Overall accuracy across all joints increases from 57% to 60%, which is a reasonable improvement given the difficulty of the data. Multiple people often interact and occlude each other, making pose estimation and motion extraction difficult.

3.5 Conclusion

We described a family of temporal features utilizing weakly stabilized video frames. Weak stabilization enables our detectors to easily extract part-centric information by removing most camera- and object-centric motion. We experimentally show that simple temporal differences extracted across large time-spans are capable of producing state-of-the-art results on the challenging Caltech Pedestrian dataset. Finally, we show our features generalize to detecting individual body parts, as well as pedestrians.

Chapter 4

N-best maximal decoders for part models

4.1 Introduction

The initial chapters of the thesis discussed people detection, with a focus on small instances. For the remainder of our the thesis, we switch our focus to the pose estimation problem, with a focus on video footage. As previously shown in Fig.1.3, combining temporal constraints (such as a hand must be the vicinity of the same hand in the previous frame) with spatial constraints (a hand must lie near an elbow) causes well-known difficulties in inference. *Tacking-by-detection* is a common alternative approach that independently applies a static spatial model to each of the frame. In this chapter, we describe a method that goes one step further and extracts multiple pose candidates per frame, but then uses a temporal model to stitch together candidates with consistent dynamics (in a sense, *tracking-by-detection and stitching.*

Most of this chapter will consider the general problem of generating multiple candidate object configurations in an image or video, within the framework of part-based models. We conclude the chapter (Sec. 4.8) by describing a tracking system that uses these candidates for articulated tracking. Though our motivating application is tracking, multiple candidate instances are generally useful strategy for resolving ambiguous image data using any form of higher-level contextual knowledge (beit temporal or otherwise). Indeed, we take our inspiration from the speech community for such an inference strategy and advocate the use of N-best algorithms for generating a set of N high-scoring candidates.

Though N-best algorithms are popular in speech, they have not been used in vision due to the fact that second-best configurations will typically be one-pixel shifted versions of the best. Crucially, one needs to enforce some form of non-maximum suppression (NMS) during the decoding process to ensure that near-identical configurations will not be returned. We describe novel and efficient appproximate N-best algorithms that return a set of putative configurations that are

1. *high-scoring*, in that they score above some user-defined threshold
2. *diverse*, in the sense that they do not overlap according to a user-defined criteria.

We demonstrate these algorithms for the problem of tracking people in video sequences. We use a recent state-of-the-art part model [100] to generate multiple pose hypotheses for each frame, and compare our approach to a variety of baselines including standard NMS and sampling algorithms. We then stitch candidates together to yield a final track, demonstrating that our pose hypotheses produce significantly more accurate tracks.

Formulation: Let us write z for a configuration of part locations, and $S(z)$ for its associated score. As in past work [17, 6], we use a simple greedy algorithm for instantiating multiple configurations: Search over the exponentially-large space of configurations z for the maximally scoring configuration, instantiate it, remove all configurations which overlap, and

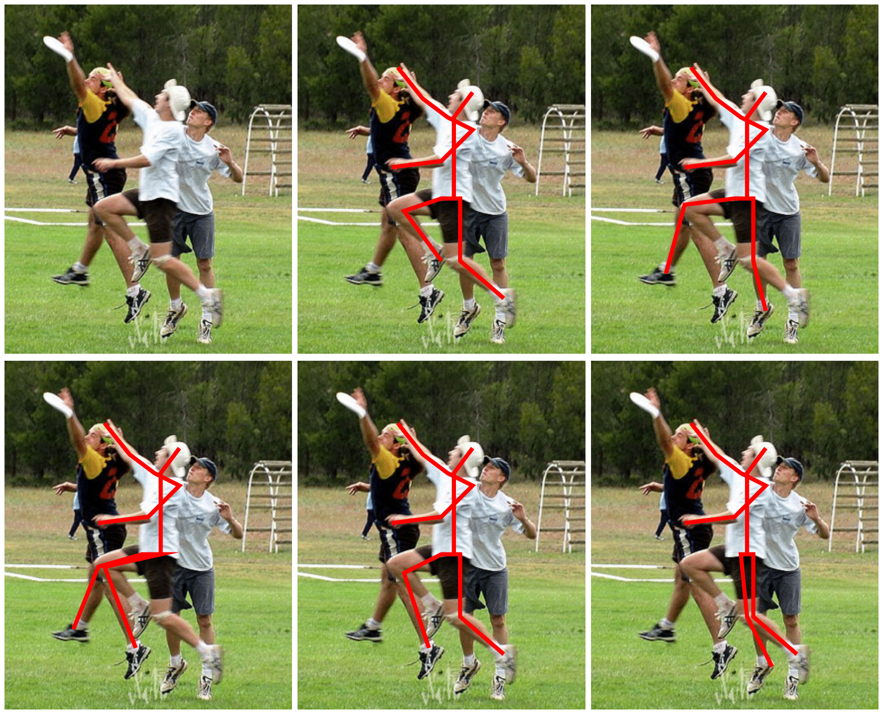


Figure 4.1: In order to localize articulated objects in cluttered scenes, one will need to reason about multiple pose hypotheses. In the above image in the **top left**, we show a true pose in the **top middle**. We show other hypotheses that may also score highly given a reasonable object model. We argue that the correct pose should be extracted from higher level contextual reasoning involving nearby objects, occlusion reasoning, etc. We describe novel dynamic programming algorithms for part-based models that can return such diverse, but high-scoring pose hypotheses from an image.

repeat. The process is repeated until the score for the next-best configuration is below a threshold or N configurations have been instantiated. A naive implementation of such an algorithm would take exponential time. If the score $S(z)$ is decomposable, one can apply a standard N-best algorithm that sequentially returns configurations [63, 102] until N non-overlapping poses are returned. We describe an approximate algorithm that is orders of magnitude faster (but near identical in performance) by exploiting decomposable notions *of overlap*.

Common approaches: It is not clear how to define overlap for configurations of multiple parts. One simple approach is to define overlap using a single "root" part; this is the approach taken in most part-models [34]. For example, one may define two human pose configurations to overlap if the root torsos overlap. This is unsatisfactory because we may still wish to consider poses with identical torsos, but different arms or legs (see Fig.4.1). Part models often make such errors due to self-occlusion or cluttered backgrounds, and one would ideally like to resolve these mistakes using higher-level reasoning (using say, temporal context). Another possibility may be to generate segmentation masks for two configurations, and then define overlap in terms of pixel overlap. However, such an approach ignores the natural semantics of body pose; consider an image of a upright person and someone performing a handstand. They may have large pixel overlap but are semantically quite different.

Our approach: We examine multiple definitions of overlap, but begin with a simple one: two poses overlap if *all* parts overlap. Under this definition, two poses that overlap for all but one part are still considered "different". This allows us to explicitly reason about poses that differ only by the location of a single part (e.g., the left hand). Under this definition and similar variants, one can compute the N-best maximal configurations by analyzing the *max-marginal* of each part. Specifically, we describe an N-best algorithm whose cost is N times the cost of computing the single-best configuration with dynamic programming. Our algorithm is approximate in that it exactly solves the formulation above only under certain conditions (which we describe), but we empirically demonstrate that it consistently produces high-quality solutions.

After discussing related work, we build the basic machinery for our N-best algorithm by reviewing algorithms for computing the best-configuration and max-marginals (Sec.4.3) in a tree-structured object model. We review an existing N-best algorithm in 4.4, and present our N-best maximal decoder in Sec.4.5. We present implementation issues in Sec.4.6, and evaluate the quality of our algorithm compared to a brute-force approach in Sec.4.7. We finally present experimental results in Sec.4.8 for video-based body pose estimation, demonstrating the superiority of our algorithm compared to standard approaches in vision.

4.2 Related work

N-best inference algorithms have been developed for chain-structured hidden markov models [64, 84], tree-structured graphical models [63], context-free grammars [44], and loopy models [102]. Though such approaches have proven effective in domains such as speech and bioinformatics, they are uncommon in vision because they tend to return pixel-shifted copies of the best configuration. We introduce N-best maximal algorithms that address these limitations by ensuring that returned configurations are non-overlapping.

Vision researchers often use sampling-based algorithms to generate multiple hypotheses for subsequent refinement. Data-driven MCMC [86] is a popular inference algorithm in this vain, which successful application to the task of body pose estimation [53, 82]. Tree-structured models have also been shown be to effective proposal distributions for evaluating non-tree scoring functions [32, 10]. We explicitly compare our method to such approaches, and show we tend to consistently generate better results. This is because our method, unlike sampling-based approaches, provides explicit control of the quality and diversity of generated hypotheses.

We illustrate our N-best algorithm for the task of tracking by stitching together N-best hypotheses from frames of a video. Such tracking-by-detection approaches are attractive because they can avoid drift and recover from errors [4, 73, 85, 38]. Exemplar-based detectors generate multiple hypotheses by finding locally maximal template responses with a coarse-scale search over poses and locations [38, 85]. These maximal responses can be refined by a local gradient search [16]. Our N-best algorithms combine these two steps by directly search over an exponentially large of configurations, using a user-defined notion of overlap to generate locally-maximal responses.

4.3 Best and next-best configurations

We write z_i for the location of part i and $z = \{z_1, \ldots, z_K\}$ for a configuration of K parts. We write $z \in Z$, where Z is the exponentially-large set of possible configurations. We score a configuration as:

$$S(z) = \sum_{i \in V} \phi(z_i) + \sum_{ij \in E} \psi(z_i, z_j) \tag{4.1}$$

where $\phi(z_i)$ is a local part score, $\psi(z_i, z_j)$ is a pairwise deformation model, often interpreted as a spring, and $G = (V, E)$ is a graph that defines relational constraints between certain pairs of parts. It is well-known that when G is a tree, one can compute $\text{Best}(Z) = \max_{z \in Z} S(z)$ with efficient one-pass dynamic programming (DP) routines that pass messages from the leaf parts to the root part[32]. By backtracking from the highest-scoring root location, one can construct the associated configuration $\text{Best}^*(Z) = \arg\max_{z \in Z} S(z)$.

We define a *marginal score* of part i at location $z_i = j$ to be the best scoring configuration

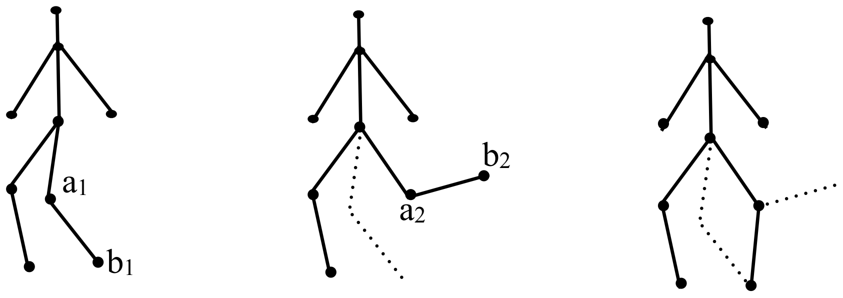


Figure 4.2: A single max-marginal table does not suffice for N-best decoding. From left to right, we show top three poses which differ by either the knee location (a_1, a_2) or foot location (b_1, b_2). Let's say the second pose was found by backtracking from the foot max-marginal at location b_2. The third pose will never be found by backtracking from any entry of the original max-marginal table. This necessitates the need for constructing constrained partitions of the configuration space.

given that part i lies at j:

$$m_Z(i,j) = \max_{z \in Z : z_i = j} S(z) \tag{4.2}$$

Standard one-pass DP already computes marginal scores for the root; these are scores which are thresholded (and possibly non-maximum suppressed) to compute a sparse set of detections in [32, 34]. To generate marginal scores for all parts, one could repeat this procedure K times, letting each part take its turn as the root. It turns out that many of the messages across these K instances are identical, and they can be implemented in an effcient two-pass DP algorithm (e.g., max-marginal inference on trees).

[102] makes the observation that the highest-entry in the max-marginal table corresponds to Best(Z), while the second-highest entry must correspond to the next-best configuration in Z. We similarly write NextBest(Z) and NextBest$^*(Z)$ for score and configuration variables of the next-best configuration. One might think that the third-best pose can be found by the third-highest entry in the table, but this is not true - see Fig.4.2. This observation is the foundation behind the iterative N-best algorithm presented in the next section.

4.4 N-best decoding

We now describe the N-best algorithm of [102] which iteratively returns configurations ordered by score. For convenience, we refer to configurations as poses. One can use this algorithm to perform N-best *maximal* decoding by repeatedly generating poses until N non-overlapping ones are returned (for any definition of overlap). As we show in Sec.4.7, this "brute-force" approach is slow because most returned poses will be overlapping. We describe an extension in the next section which is orders of magnitude faster for decomposable notions of overlap.

The algorithm works by iteratively partitioning Z into N sets, such that the best pose for each set is one of the N-best. Initialize the first set to be the entire set of configurations $Z_1 = Z$, and compute the best pose $c^1 = Best^*(Z_1)$. Iterate the following for $t = 2 : N$:

```
1 $(i', j', n') = \arg\max_{i,j,n<t} \mathrm{NextBest}(Z_n)$
2 $c^t = \mathrm{NextBest}^*(Z_{n'})$
3 $Z' = \{z : z_{i'} = j'\}$
4 $Z_t = Z_{n'} \cap Z'$
5 $Z_{n'} = Z_{n'} \setminus Z'$
```

The final set of N-best poses is $\{c^1, \ldots, c^N\}$. We refer the reader to [102] for a detailed proof, but provide a visualization of the algorithm in Fig.4.3.

4.5 N-best maximal decoding

We now show how one can modify the presented algorithm to directly return poses that are diverse by exploiting decomposable notions of overlap. We define two poses $z^1, z^2 \in Z$ as

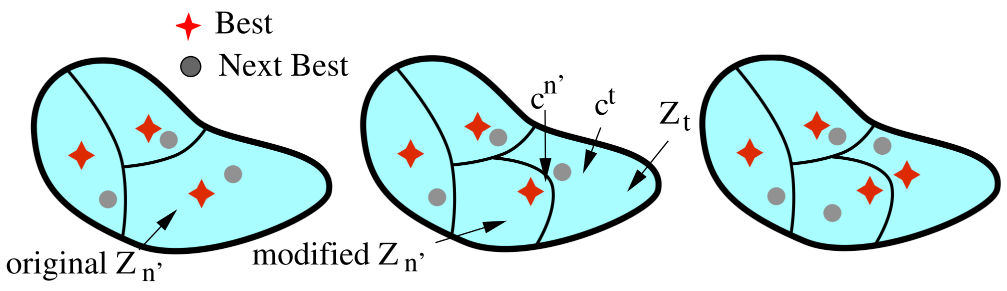


Figure 4.3: We visualize the iterative N-best decoder of [102], described in Sec.4.4. Assume we are at the begining of iteration $t = 4$. We have already partitioned Z into 3 sets such that the Best pose in each make up the best 3 poses (**left**). The 4^{th}-best pose must lie in some set $Z_{n'}$ (because the partitioning covers Z) and must be equal to NextBest($Z_{n'}$) (by the definition of NextBest). Lines 1 and 2 of the algorithm find this next best pose where we write (i', j', n') for the index of the max-marginal table entry and partition index where it was found. Lines 3-5 further paritions $Z_{n'}$ in two (**middle**) such that we now have a 4-set partitioning such that the Best of each make up the top 4 poses (**right**).

overlapping if each part overlaps:

$$\text{ov}(z^1, z^2) = \bigwedge_i \text{ov}_i(z_i^1, z_i^2) \tag{4.3}$$

where ov_i is a symmetric predicate for defining overlap of individual parts. One may define two parts as overlapping if the area of their intersection exceeds 50% of the area of their union - this is benchmark criteria used in PASCAL [29]. Alternatively, for articulated parts, one may use the endpoint-error criteria common in pose estimation benchmarks [36]. In pose-based action recognition, it may be important to reason about poses with different end effector locations (e.g., hands and feet). We can do this with a part-specific overlap relation ov_i.

The following lemma states that one can find the next-best non-overlapping pose by examining the max-marginal table:

Lemma 4.1 *Given a set of poses Z and their associated max-marginals $m_Z(i, j)$, the score* *of the next-best pose that does not overlap $Best(Z)$ is:*

$$NextOvBest(Z) = \max_{i,j:\neg ov_i(c_i,j)} m_Z(i,j) \tag{4.4}$$

Proof The next-best non-overlapping pose must contain at least one part i that does not overlap c_i. The max-marginal table allow us to enumerate each possible part and non-overlapping location.

To use the partitioning approach of the previous algorithm, we need to add an additional constraint to ensure that a partition is valid with respect to overlap:

Lemma 4.2 *Let $\{Z_n\}$ be a partitioning of Z that satisfies the following condition.*

$$\neg ov(NextOvBest^*(Z_n), Best^*(Z_m)) \quad \forall n, m \tag{4.5}$$

We call such a partitioning **non-overlapping**. *The score of the next-best configuration that does not overlap any $Best^*(Z_n)$ is:*

$$\max_n NextOvBest(Z_n) \tag{4.6}$$

Proof Because $\{Z_n\}$ partitions Z, the next-best configuration must lie in $Z_{n'}$ for some n'. If it is not NextBest(Z_n'), then there exists another higher-scoring configuration which does not overlap any Best$^*(Z_n)$. This is a contradiction of Lemma 4.1.

We now can describe our N-best maximal decoder. Initialize Z_1 and c^1 as in Sec.4.4, and iterate the following for $t = 2 : N$:

The N-best algorithm from Sec.4.4 is a special case of the above algorithm obtained by defining a single-pixel overlap predicate $\text{ov}_i(z_i^1, z_i^2) \Leftrightarrow (z_i^1 = z_i^2)$. The main differences

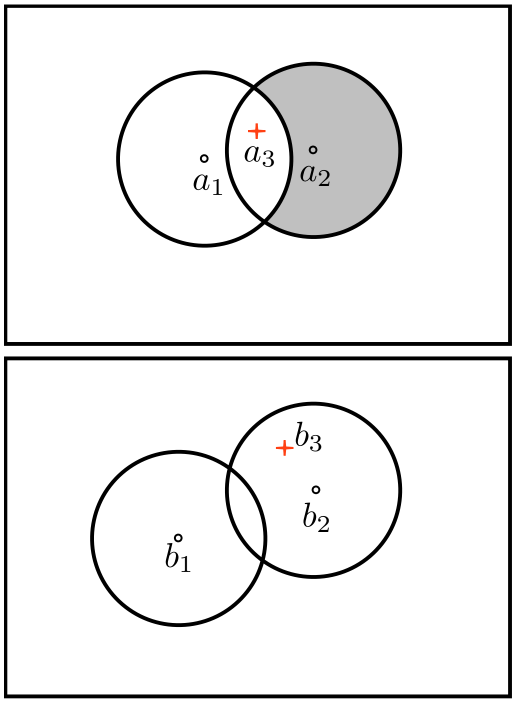
```
1 $(i', j', n') = \arg\max_{i,j,n<t} \text{NextOvBest}(Z_n)$
2 $c^t = \text{NextOvBest}^*(Z_{n'})$
3 $Z' = \{z : \text{ov}_i(z_{i'}, j') \wedge \neg\text{ov}_i(z_{i'}, c_{i'}^{n'})\}$
4 $Z_t = Z_{n'} \cap Z'$
5 $Z_{n'} = Z_{n'} \setminus Z'$
```

are two fold: the NextBest function is replaced by NextOvBest, and Step 3 is refined to ensure that that z'_n is sub-partitioned into two sets who's Best poses do not overlap. If the NextBest poses are also nonoverlapping, than one can invoke Lemma 4.2 to ensure that at the next iteration, the algorithm will find the true next-best non-overlapping pose. If not, the next iteration will return a pose that overlaps with one of the previously-returned poses. In practice, we find that Lemma 4.2 holds the vast majority of iterations, implying that our algorithm (usually) returns the optimal set of poses. We show a failure case in Fig.4.4. In this case, one could simply ignore such invalid poses, and continue iterating until N non-overlapping poses have been found. We present a further analysis of such errors in Sec.4.7.

Hybrid decoder: We can turn the above algorithm into an optimal N-best decoder by identifying the faulty sets that violate Lemma 4.2, and resorting to the brute-force N-best algorithm from Sec.4.4 when refining those sets. This can be implemented by changing the overlap function ov_i to use single-pixel overlap when considering poses within such sets. In Sec.4.7, we contrast the performance and speed of this hybrid algorithm versus the brute-force and approximate algorithm.

4.6 Efficient implementation

Representing partitions: One needs an implicit representation for each partition Z_n, since one cannot directly enumerate such exponentially-large subsets. We represent each partition with a set of quadruples $\{(i', j', c_{i'}^{n'}, y')\}$ where $y' \in \{0, 1\}$ is a bit that specifies whether or



Iteration 1:

$\begin{bmatrix} a_1 & b_1 \end{bmatrix} = \text{Best}^*(Z)$

$Z_1 : \{\}$

Iteration 2:

$(a, a_2, 1) = \underset{i,j,n \in \{1\}}{\text{argmax}} \, \text{NextOvBest}(Z_n)$

$\begin{bmatrix} a_2 & b_2 \end{bmatrix} = \text{NextOvBest}^*(Z_1)$

$Z_2 : \{(a, a_2, a_1, 1)\}$

$Z_1 : \{(a, a_2, a_1, 0)\}$

Iteration 3:

$(b, b_3, 1) = \underset{i,j,n \in \{1,2\}}{\text{argmax}} \, \text{NextOvBest}(Z_n)$

$\begin{bmatrix} a_3 & b_3 \end{bmatrix} = \text{NextOvBest}^*(Z_1)$

$Z_3 : \{(a, a_2, a_1, 0), (b, b_3, b_1, 1)\}$

$\vdots$

Figure 4.4: We illustrate the first three iterations of our algorithm for a two-part (a and b) model. On the **left**, we show part detections, and designate a region of overlap around every detection with a circle. The second pose, (a_2, b_2), is obtained by backtracking from the max-marginal entry a_2. We partition Z into two sets such that (a_1, b_1) is the best pose in Z_1, and (a_2, b_2) is the best pose in Z_2, where Z_2 is the set of poses that overlap a_2 but don't overlap a_1 (the shaded region). We represent sets using quadruples, as explained in Sec.4.6. Assume the next-best max-marginal entry is found in marginal location b_3, in set Z_1. It is possible that the backtracked pose (a_3, b_3) might overlap (a_2, b_2), shown in red. We show in Sec.4.7 that this is a rare occurence.

not part i overlaps region R, where R is the set of locations that overlap j and do not overlap location $c_{i'}^{n'}$ (Fig.4.4). Each quadruple represents a constraint that is iteratively added as the algorithm adds next-best configurations and partitions the set $Z_{n'}$ from which they were found.

Memory: As written, the above algorithm requires storing and searching over N max-marginal tables at Step (1). We need to store only the best and next-best configuration for each partition, together with the part index i' and location j' that triggered the next-best configuration. Hence we can compute max-marginal tables *in place*: once we create a new partition (in Step 4 and 5), we compute the best and next-best configurations for each. We can then safely ignore its max-marginal table. This means each iteration of the algorithm requires 2 max-marginal computations, making our overall N-best algorithm linear in N (as in [102]).

Caching: We compute local part scores $\phi(z_i)$ from (4.1) once, and reuse them to compute max-marginals for any given partition. This can be done by temporarily invalidating part scores for locations outside a partition, and running the two-pass max-marginal algorithm from Section 4.3. If we assume the deformation model $\psi(z_i, z_j)$ from (4.1) is bounded, one can limit the amount of max-marginal computations that must be updated at each iteration. Say, for example, that the head and leg part can be at most δ pixels apart. This means that, if we add the constraint that heads must (not) overlap a particular location, we need recompute max-marginals only for parts that lie within δ pixels of the head location. This can be efficiently implemented by computing a distance transform over a small δ-radius sub-window in an image, rather than the entire image.

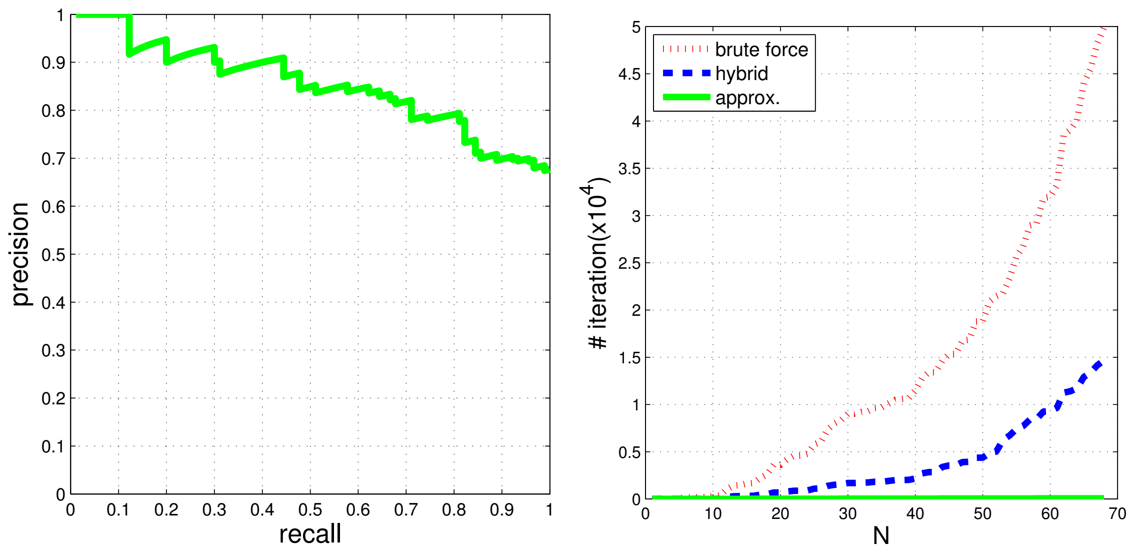


Figure 4.5: Quality and speed of approximation. The curve on **left** shows the accuracy of the our algorithm. We use top 90 non-overlapping poses of a reference image for evaluation. We achieve high accuracy over the entire range of recall rate (**AP = 85.4%**). On **right**, we show the number of iterations of each algorithm required to find N non-overlapping poses. Our algorithm takes 87 iterations to generate 68 poses, while brute force and hybrid approaches take about 50k and 15k iterations.

4.7 Analysis of approximation

We compare the accuracy and speed of the brute-force N-best maximal algorithm (Sec.4.4), as well as our approximate and hybrid algorithm (Sec.4.5) on a random reference image. As we run our iterative algorithm for $t = 1 \dots N$, we count the fraction of poses which are present in the top t optimal results, scoring both the precision (the fraction of poses we return that are optimal) and recall (the fraction of the optimal poses we return). We also compare the speed of each algorithm by counting the iterations needed to obtain t non-overlapping poses. Our approximate algorithm is faster than brute force and hybrid approaches by three orders of magnitude, while generating almost the same result.

4.8 N-best tracking results

We demonstrate our algorithms by applying them to the problem of tracking people in video sequences. We generate candidates from each frame of a video, and stitch them together with dynamic programming. We use the recent articulated part-based model of [100], which appears to be the current state-of-the-art system as evidenced by various pose-estimation benchmarks. We demonstrate that, even given this high-accuracy detector, locally ambiguous hypotheses can be refined by exploiting temporal context from neighboring frames.

Temporal context: Assume for frame t in a video, we generate N candidate poses. Let $k_t \in \{1, \ldots N\}$ be a pointer to a particular pose. We wish to maximize the score:

$$\text{Score}(k) = \sum_t \text{Local}(k_t) + \alpha \text{Pairwise}(k_t, k_{t-1}) \tag{4.7}$$

where $\text{Local}(k_t)$ is the score of candidate pose k_t computed by (4.1). We write $\text{Pairwise}(k_t, k_{t-1})$ for an arbitrary pairwise term penalizing the difference of two configurations. In practice, we simply use the (negative of the) total squared pixel difference between each joint in pose k_{t-1} and pose k_t. We also experimented by penalizing the change in appearance of parts, and saw a minimal improvement in accuracy. The parameter α controls the trade-off between the two terms, and was tuned manually. The above score can be optimized by standard dynamic programming on a trellis graph.

Algorithms: We compare our approach of generating N-best candidates with several baseline algorithms for generating N candidates. The simplest is **noNMS**, which perform standard 1-pass dynamic programming, but then backtracks from the N top-scoring root marginals to generate N candidates. As one might suspect, the N candidates tend to be pixel-shifted versions of each other. We also consider **rootNMS**, which performs NMS on the root scores to avoid returning pixel-shifted root locations. We applied **noNMS** to find a very large set of candidates, and then post-processed them to find the best N configurations that do not overlap according to definition (4.3); we denote this baseline as **partNMS**. Finally, we also compare to the sampling baseline advocated in [32, 10]. In particular, we use the max-marginal sampling algorithm **MMsampling** of [10], which seems to be the current state-of-the-art approach for generating multiple samples from a part model. The sampler requires a temperature parameter that loosely controls the amount of diversity; we found results were sensitive to this parameter and put forth considerable effort to tune it.

To illustrate the ability of our approach to handle user-defined overlap functions (4.3), we compare two versions of our algorithm. We write **Nbest(all)** to denote an overlap function which treats all parts equally, where two parts are defined as overlapping if their bounding boxes intersect at all. We write **Nbest(limb)** to denote an overlap function that only requires leaf parts (hands, heads, and feat) to be non-overlapping. This can be implemented by defining ov_i to be 1 for all non-leaf parts, regardless of their position z_i.

Evaluation: We assembled a set of video sequences with varying degrees of clutter (Fig.4.6) [73, 81]. We quantitatively evaluate our algorithms in two ways; we look at the overall track score from (4.7), and we evaluate tracking accuracy using the now-standard *Percentage of Correct Parts* (PCP) criteria introduced in [36]. To perform the latter, we manually annotated ground-truth limb locations in these sequences. We will make these annotations publicly available to spur further quantitative evaluation.

Analysis: We show qualitative results for various algorithms in Fig.4.7. We refer the reader to the caption for detailed analysis, but note that our algorithm consistently produces more diverse and higher quality hypothesis than standard approaches. We present PCP results in Fig.4.9. We refer the reader to the caption for a detailed analysis, but our N-best algorithm consistently outperforms all baseslines. In general, both our approach and sampling do much better than the baseline NMS algorithms. We further analyze this behaviour in Fig.4.10, and show that for small N, our approach clearly outperforms sampling because we are guaranteed

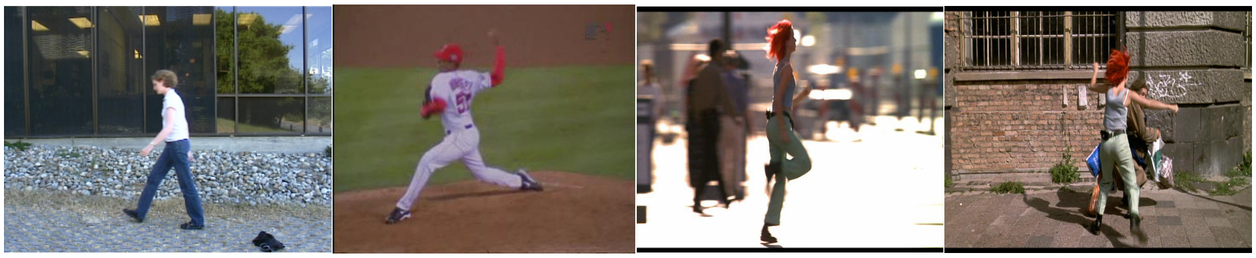


Figure 4.6: We use four video sequences for evaluation, used in previous work [73, 81]. From left to right, we name them as **Walking**, **Pitching**, **Lola1**, and **Lola2**. They exhibit varying degrees of clutter (including multiple people), camera movement, and body poses.

to report high-scoring configurations while a sampler is not.

Computation: We have implemented our algorithm with a subset of caching speedups proposed in Sec.4.6. For small $N < 10$, our algorithm is similar in speed to the baselines above. For large N, our linear dependance on N dominates the effect of our caching, making our approach slower than the baselines. We are exploring alternate approximate algorithms that further sacrifice some performance for speed.

Conclusion: We have described a general method for returning back N configurations from a part model that do not overlap, according to some user-defined notion of overlap. We show that our algorithm produces, both qualitatively and quantitatively, a strong set of hypotheses that can be used for subsequent refinement using more complex, intractable objective functions. We believe our N-best formalism provides a practical and general approach for minimizing such complex functions, similar to such inference strategies from the speech recognition community. As suggested in Fig.4.10, there still remains a disconnect between objective functions currently in use and overall accuracy, and so we are currently pursing approaches for learning meaningful objective functions from data using N-best decoders.

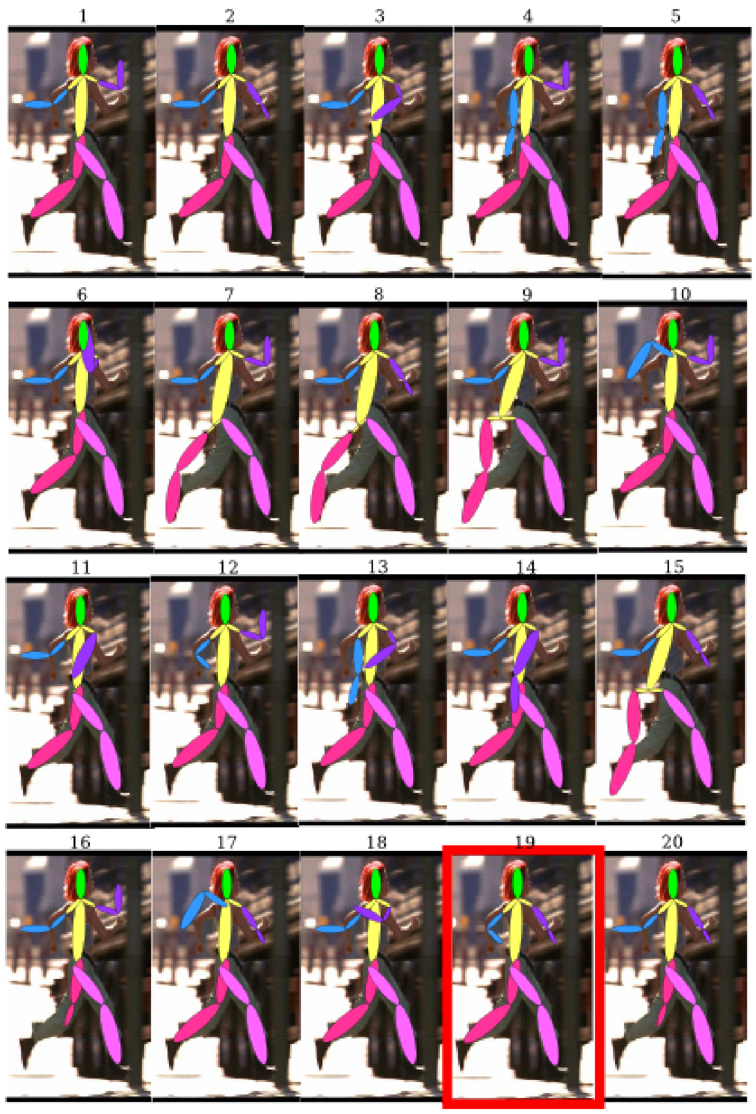


Figure 4.7: We show the **20-best** configurations returned by our N-best algorithm for a frame in the *Lola* video. Note that each configuration contains at least one part that does not overlap any other configuration. Since there exists arm-like clutter at the top of the image, many of the top-scoring hypotheses consider various arm positions. Note that many of these configurations share the same root; hence they would not returned from typical NMS-based algorithms for generating multiple detections in an image. We show final configuration selected by the DP tracker in **red**, which was the 19-th returned pose.

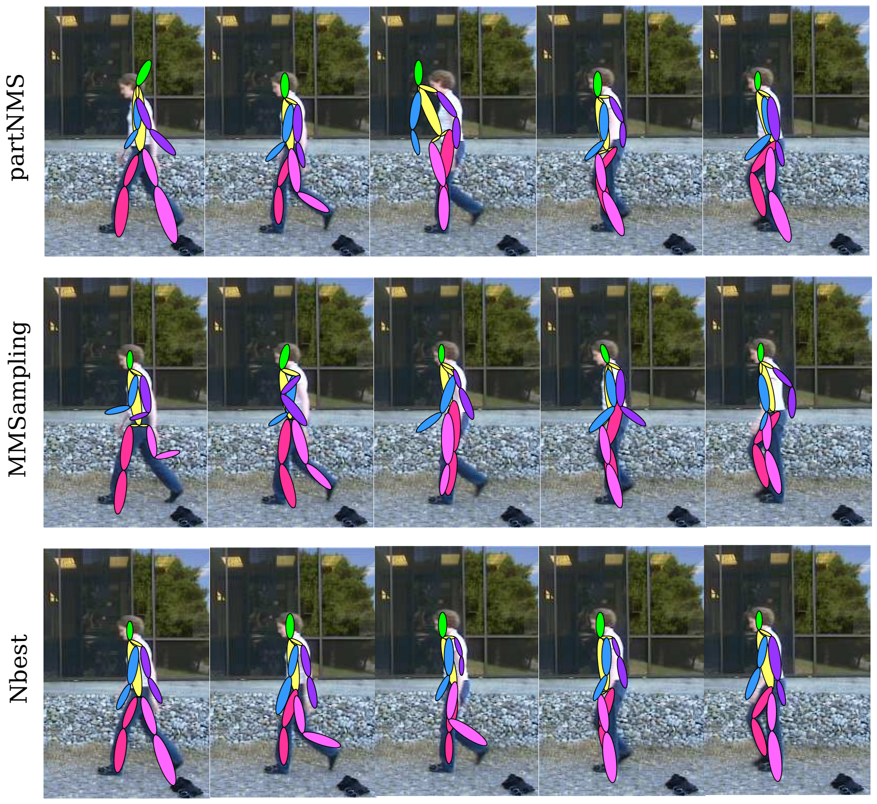


Figure 4.8: Tracking result for the *Walking* sequence. **partNMS** tends to estimate wrong head(**first and third**) frame, because it can only report a single configuration for the best root part. **MM sampling** tends to report noisy samples with varying degrees of quality due to its stochastic nature. Due to the looseness of the PCP scoring criteria, we found that many of these configurations were scored as correct, though qualitatively they appeared to be noisy. **Nbest** tends to generate reasonable looking results.

Algorithms	walking	pitching	lola1	lola2
noNMS	0.825	0.762	0.505	0.445
rootNMS	0.815	0.741	0.455	0.390
partNMS	0.825	0.762	0.515	0.420
MMsmpl	0.930	0.800	0.645	0.440
Nbest(all)	0.940	**0.800**	0.635	0.495
Nbest(limb)	**0.950**	0.797	**0.670**	**0.500**

Figure 4.9: We compare average PCP of tracks derived from N=300 candidates for baselines and our algorithm. Our approaches dominate all baselines, including the state-of-the-art method of [10]. We further analyze the behaviour of all algorithms in Fig.4.10

.

Figure 4.10: On the **left**, we show PCP accuracy as a function of N, the number of generated hypotheses, for various algorithms. Most algorithms tend to produce stable tracks for $N > 100$. We examine their behaviour over the first $N < 50$ generated hypotheses in the **middle**. In general, we see that our N-best algorithm tends to produce accurate tracks, even for small N. Rather than scoring tracking accuracy, we can score the ability of various algorithms to maximize the objective function from Eq.4.7 (on the **right**). We see that our algorithm consistently produces better scores than sampling, particularly for small N. This makes sense since for $N = 1$, our algorithm reports back the overall best configuration in a frame, while sampling algorithms may report back (in theory) any configuration. In general, the disconnect between the **right** and **middle** plots suggests that algorithms that perform better at maximizing our objective function may not produce better tracks. This indicates, that in addition to our focus of better inference algorithms, we still need better objective functions to maximize.

Chapter 5

Exploiting synthetic video frames for pose estimation

5.1 Motivation

In the previous chapter, we explored combinatorial N-best methods for exploiting temporal constraints in articulated tracking. In this chapter, we explore the alternate dimension of building better pose candidates through tuned appearance models.

Consider the problem of tracking human poses in a one-take video clip, say "Phoebe" in "Friends" TV show. Intuitively, given the very first frame of the video, humans can immediately picture in mind what the other frames might look like; Phoebe, who was folding arms in the first frame, may be reaching her arm to grab a cup or be answering the phone in the other frames. In addition, the plate of salad might become empty; the ketchup bottle on the table might be in her hand or moved to the other side of the table. What we *don't* expect to see in the future frames is that Phoebe suddenly change her skin color or becomes noticeably fat, or that the background scene abruptly changed from a restaurant to a gym.

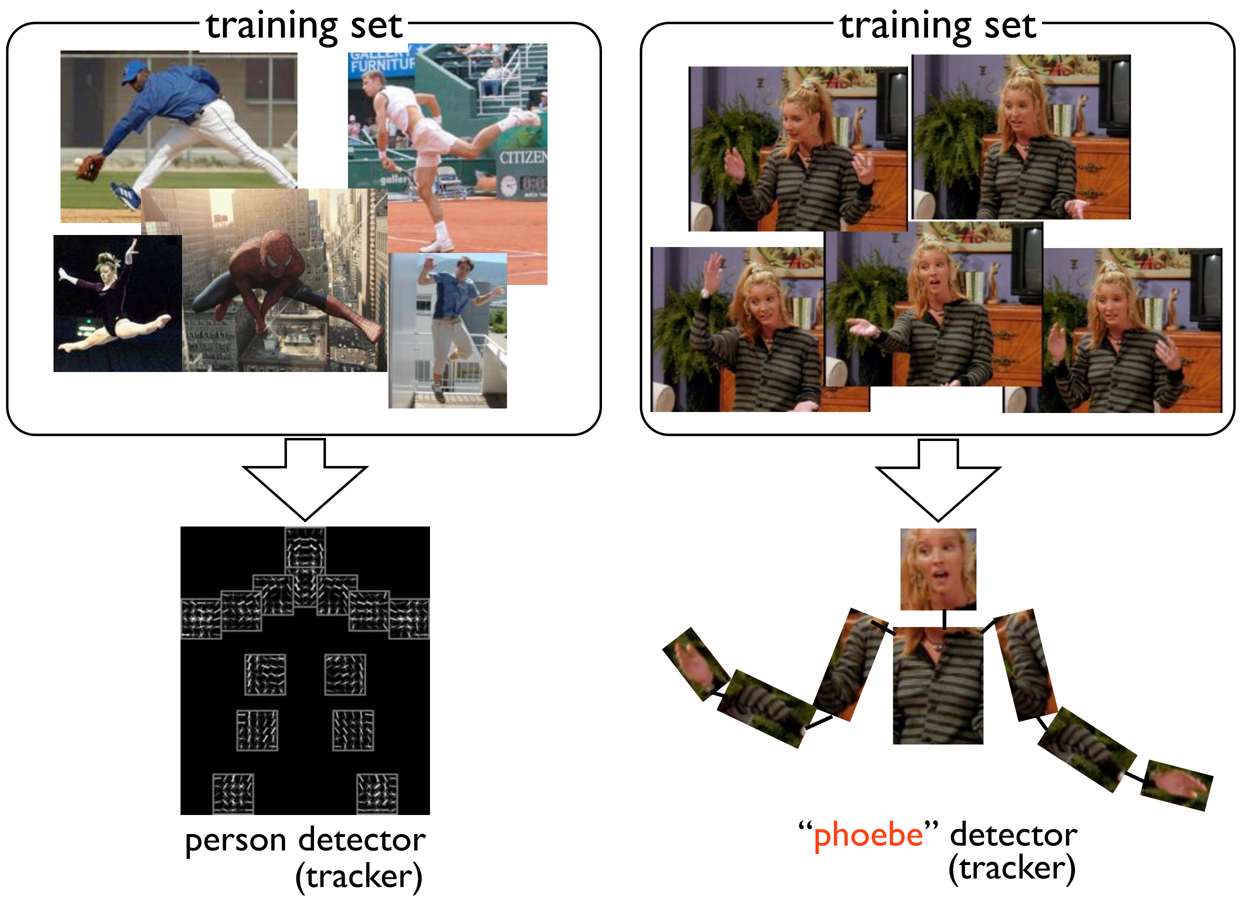


Figure 5.1: *Overfit the video!* We propose to use synthetic video frames that emulate hypothetical test frames as training data for performing recognition in video. Previous approaches use as base models generic detectors trained using images in the wild (**left**). We show that, by using training data customized to a particular video (**right**), one can achieve state-of-the-art performance on challenging pose estimation problem even with simple models and features.

On the other hand, the standard method to train a static pose detector as a basis for tracking poses in videos is quite counter-intuitive. The base detector is a generic pose detector, which is usually trained using images of arbitrary persons and background, as well as arbitrary poses. This seems to be an overkill, since it makes the training strictly harder by forcing the detector to recognize Obama's pose in the White house even though the detector would only see Phoebe in her living room.

In this chapter, we propose to train a highly-tuned appearance model to *overfit* the particular video using a large set of synthetic training data, given an annotated first frame of the video. We use a custom training data that emulate all possible future frames one expect in the given video. Unlike training data for generic pose detectors, the dataset is invariant in, e.g., clothes, body shapes, skin colors, or hair styles of people (Fig.5.1). There also exist large invariance in background. In theory, one can generate a set of reasonable custom background by modeling the scene and dynamic objects.

The most interesting mode of variation in the dataset is human pose. Therefore, our custom data covers all kinematically possible upper body poses of given character. Given an annotated first frame as a seed frame, we generate a large number of hypothetical test frames with different poses, using standard image-based rendering algorithms. Other modes we synthesize include locations and scale of humans, dynamic objects (e.g. cups, chairs) in the scene, and a small degree of background translation.

It is also interesting to notice that the expected video frames in our imagination are only in moderate detail: humans cannot predict how the wrinkle of right collar change when the Phoebe raise her left hand. In fact, it is unrealistic to synthesize all possible frames that are different in such fine detail, since the size of the dataset grows exponentially with respect to the number of parameters in the synthesis engine. A simple solution is to render low resolution frames. This allows us to synthesize only the frames that are distinct after blurring.

To predict a pose, one can cast our custom training data to any sort of feature transformation and learning algorithm to train a pose estimator. However, it is reasonable to expect the learning must be greatly simplified, since the data effectively lives in a very small subspace compared with images in the wild. We verify this idea by training pose estimators using extremely simple features and non-parametric learning algorithms, such as *raw pixel values* and *nearest-neighbor regressors*.

Overview of chapter After discussing related work in Sec.5.2, we describe our synthesis engines in Sec.5.3. Pixel synthesis is the process of generating a synthetic frame given a target pose and pose-annotated first frame; pose synthesis defines a set of target poses to synthesize. In Sec.5.4, we describe a simple nearest-neighbor algorithm for estimating pose. In Sec.5.5, we diagnose and evaluate our approach for the task of estimating upper body pose using *Friends* dataset.

5.2 Related work

Visual tracking: The problem of visual tracking have been addressed in various settings of inputs and initializations; first-frame labeled [103], online tracking [97], interactive tracking [39], etc. Articulated tracking [39], [106], [76] recently gained attention. (See [83] for complete discussion.) Especially, tracking with learned appearance models [73, 47] have proved to be effective. Our work is closest to [49] in that they use labeled first frame to track articulation of human body. Our work is also well aligned with [73], [106] in that they attempt to use consistency existing throughout the video. But our work is unique in that we focus on training data to exploit consistency without using temporal cues.

Layered shape models: Since the pioneering work of [93], layered shape model as a week form of a 3D model has been widely accepted as a useful representation for image formation.

Our work is closely related to [99], where the authors introduce generative probabilistic models that formulate layered models for object detection and segmentation.

Synthetic training data: There exists a steady body of work that has examined pose estimation using (partially) synthetic training data. Perhaps the earliest example dates back to [79], who use a large set of rendered poses for nearest-neighbor (pose) regression. [52] generate synthetic rendering of real objects under synthetic backgrounds, using green-screening. The recent work of [45] has generated a 3-million frame dataset of synthetic images of 3D articulated models in real backgrounds. Our work differs in that we perform "image-based rendering", cutting and pasting existing images to yield novel ones. From this perspective, our approach is most related to [72], who fit 3D articulated models to real images, and generate synthetic renderings by slightly perturbing joint angles. However, in our case, we do not need to synthesize appearance variations since we want to train a model tuned for the appearance of a particular figure in the video.

5.3 Synthesis engine

At the heart of our approach is a simple 2D synthesis engine that artificially generates hypothetical video frames, given the labeled first frame of the video. Unlike other synthesis-based approaches which produce high-quality synthetic images ([72], [45]) our goal is to produce a large set of *reasonably* photorealistic images which captures most of the variability expected in the future frames. Most importantly it captures various poses, but also locations, scale, camera movement, and other dynamic objects in the scene. The synthesis process consists of two components: pixel synthesis and pose synthesis, which are discussed in the following subsections. Fig. 5.2 summarizes the overall process.

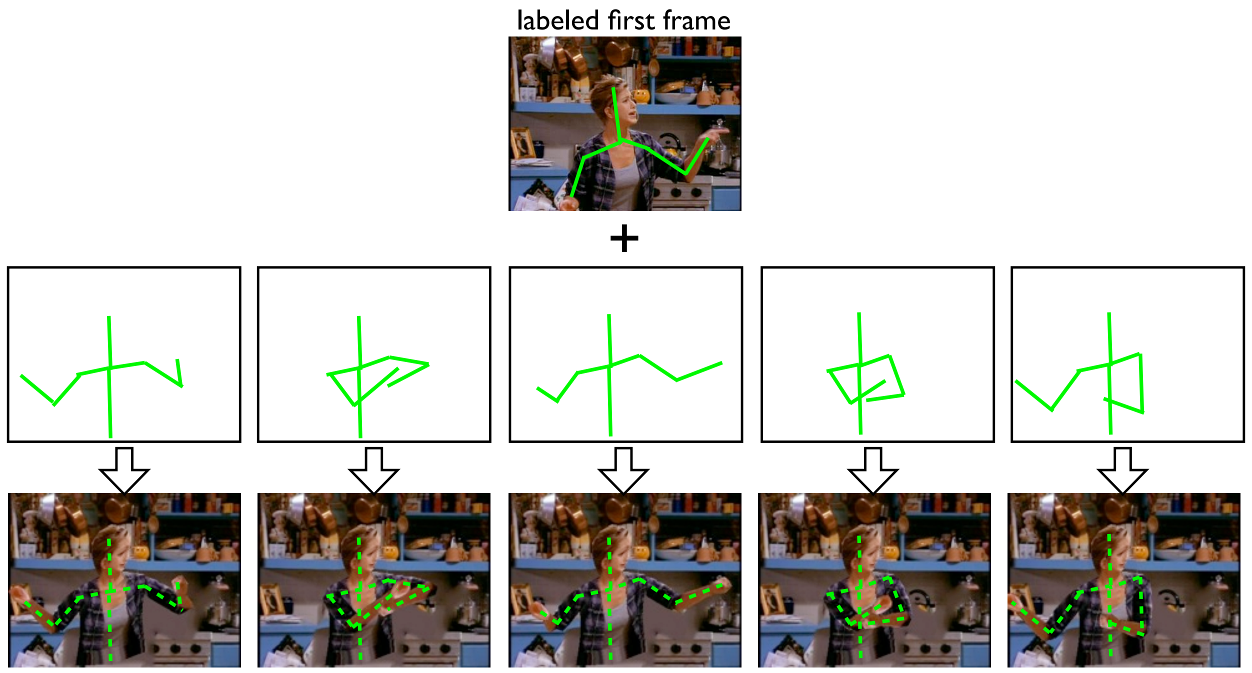


Figure 5.2: Synthesis overview. We assume the first frame of the video, as well as its pose label, is available upon test time (**top**). We define a set of poses that represents all potential poses in the video. (**middle**). We use an image-based rendering engine to warp the body parts of the first frame into the target pose. We use the collection of rendered frames to construct a custom "Rachel" training data as shown in Fig. 5.1

Pose parameterization Each part p is represented by a 4-tuple: its orientation, foreshortening ratio, and depth layer (θ, r, d). A pose $\mathcal{P}$ is defined by its root location in the image coordinate, scale, and a collection of 9 upper body parts: left/right upper arms, lower arms, hands; and torso, neck, face. That is, $\mathcal{P} = (x_0, y_0, s, (\theta, r, d)_{1:N})$. When combined with a Pictorial Structures (PS) model, this parameterization is readily converted to a skeleton representation, $(x_1, y_1, x_2, y_2)_{1:N}$, visualized in Fig. 5.2. The advantage of $\mathcal{P}$ is that it can disambiguate the difference between "short arm" and "foreshortened long arm".

Assumption As in [49] and [103], our synthesis process assumes that the first frame of the video is given with a human annotation. The annotation is in the skeleton format shown in Fig. 5.2, which require users to provide 9 body joint keypoint locations and their depth layers. From this annotation, we derive the pose parameters $\mathcal{P}_0$ with a straightforward heuristic to decide scales of parts.

5.3.1 Pixel synthesis

The pixel synthesis engine takes the labeled first frame, $(I_0, \mathcal{P}_0)$ and a target pose $\mathcal{P}_q$ as input, and produces a synthetic video frame I_q. We illustrate this process in Fig. 5.3. It is based on a 2.1D representation of image ([99], [93]), where each pixel is augmented by its depth layer. To derive a part region from pose parameters (s, θ, r, d), we need to define a kinematics model and a shape model.

Forward kinematics. *Local* pose parameters defined in $\mathcal{P}$ are relative to its parent part (e.g. lower arm bent by 30 degree at elbow). Forward kinematics ([70], [60]) is a standard way to recursively derive a *global* pose in *image reference frame* (e.g. lower arm aligned with y-axis of image) from local pose parameters. Specifically, given $\mathcal{P}$ and a Pictorial Structures (PS) model, we derive two global pose parameters for a part indexed by i; orientation angle θ_i^g and translation vector t_i^g. t_i^g is the location of joint (e.g. elbow for lower arm) in image coordinate. For the time being, we ignore foreshortening effect.

$$\theta_i^g = \theta_{par(i)}^g + \theta_i \tag{5.1}$$

$$t_i^g = t_{par(i)}^g + R(\theta_{par(i)})l_i \tag{5.2}$$

$$\theta_{root}^g = \theta_{root} \tag{5.3}$$

$$t_{root}^g = (x_0, y_0) \tag{5.4}$$

where $par(i)$ is the part index of parent of i as defined in PS, and l_i is the location offset of joint with respect to the joint of its parent (e.g. default location of elbow in the reference frame of shoulder). In case that l_i aligns with the direction that foreshortening occurs, $l_i r_i$ is used as the offset.

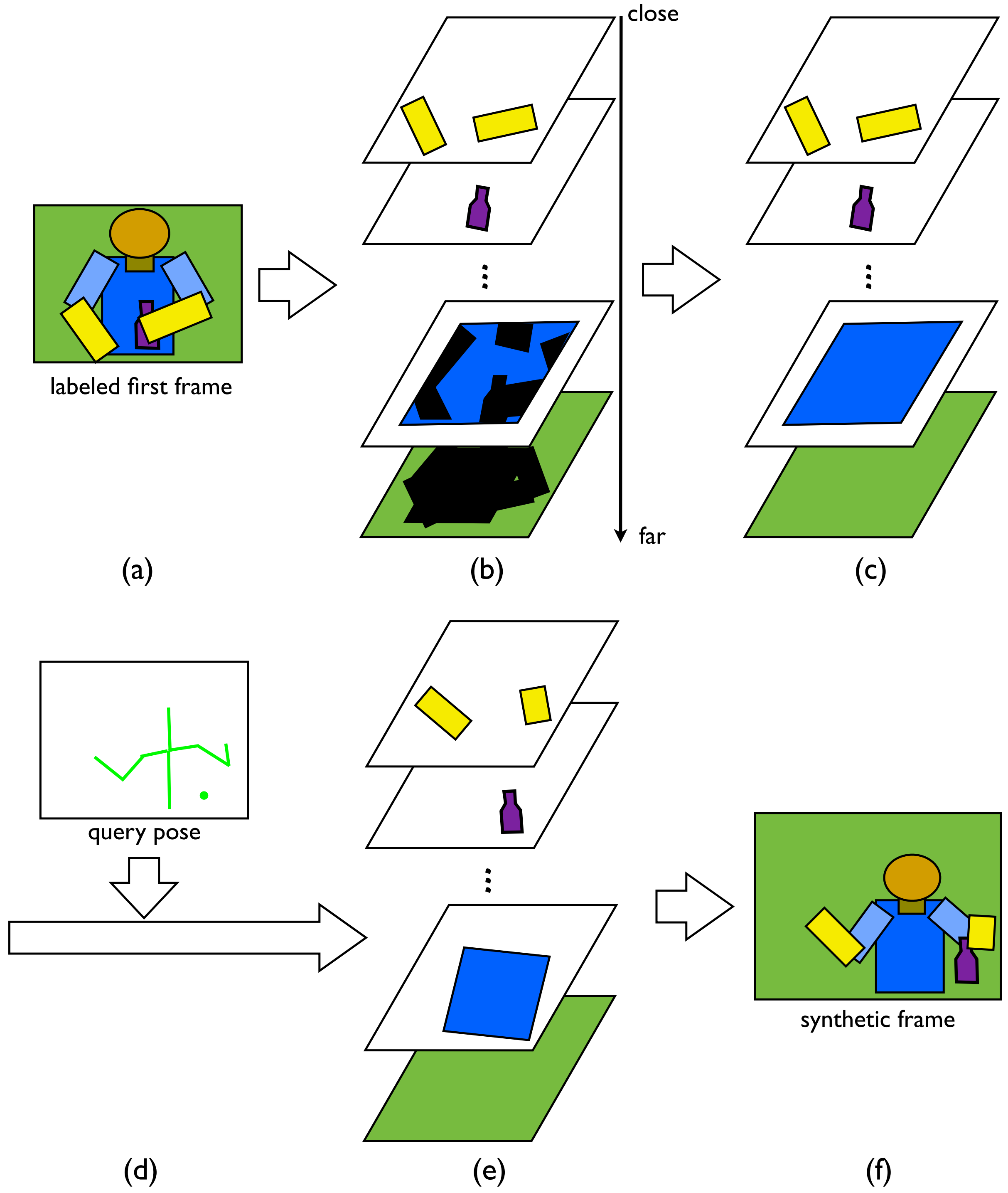


Figure 5.3: Pixel synthesis. Our rendering engine is based on a 2.1D representations of images. We first decompose the labeled first frame **(a)** into multiple layers according to their depths **(b)**. The unkown pixels due to occlusion (black region) is estimated using standard hole-filling algorithms.**(c)**. Given a target pose **(d)**, we warp corresponding body parts in (c) by rotating and/or rescaling **(e)**. By overlaying each layer according to target depth orders, we produce a synthetic frame **(f)**.

As a result, we can convert $\mathcal{P}$ to a global pose paremeterization:

$$\mathcal{P}^g = (s, (t^g, \theta^g, r, d)_{1:N}) \tag{5.5}$$

Shape model. A shape model takes (t^g, θ^g, r) of a part and s, and return a part mask M_i. Typically, M_i is represented by a polygonal contour that defines a shape ([5], [26], [49]). In this chapter, we consider *mean shape model* obtained by averaging annotated contours of 5 different characters in the dataset (Fig.5.4).

The first step of pixel synthesis is to decompose the first frame into multiple layers that are ordered according to its depth. Using the annotated pose and a shape model, we extract a pixel region for each part M_i^0. Each region is divided into two types of subregions: foreground regions that actually compose the given part, and undetermined regions of which pixel values are unknown due to occlusion by other parts. Most layers include undetermined regions. We estimate the unknown pixel values using standard hole-filling algorithms such as texture synthesis [7], linear interpolation, pixels from the symmetric part or the same part. We denote R_i^0 as RGB pixel values of M_i after the hole-filling.

Given a target pose $\mathcal{P}_t$ (and therefore $\mathcal{P}_t^g$), we first derive a target mask for each part M_i^t, and generate corresponding RGB pixels R_i^t using R_i^0. We use standard image-based rendering techniques to find a local affine warp from M_i^0 to M_i^t.

Finally, by overlaying layers according to the target depth assignments, we produce a synthetic frame of the target pose. If parts have been ordered from front ($i = 0$) to back ($i = N$), the final rendered image is generated as following:

$$I^t = I_0 \quad \text{where} \quad I_i = (1 - M_i^t) I_{i+1} + M_i^t R_i^t \tag{5.6}$$

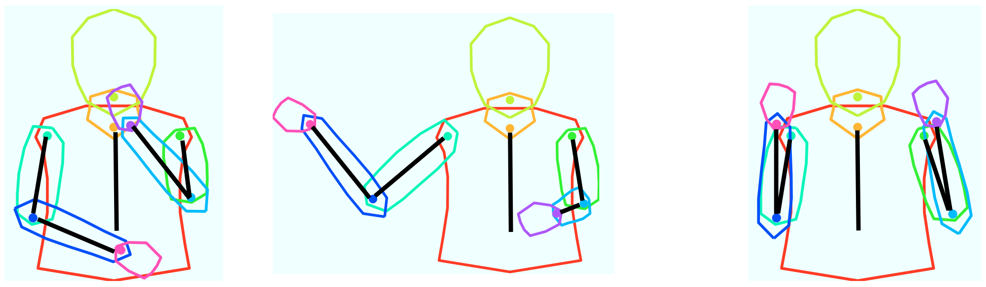


Figure 5.4: Upper body shape model. The 2D shape of each body part is a polygon parameterized by scale, orientation, and forshortening ratio, which are mostly represented by pose skeletons (black).

5.3.2 Pose synthesis

In this section, we describe how we define a set of poses to synthesize. The general rule is to synthesize all possible video frames one expects to see given the first frame of the video. Ideally, we would synthesize all possible poses, locations and scales of the human, all possible camera translations, and all possible dynamic scene elements. We make simplifying assumptions that the video is stabilized, and that the central figure and interacting objects are the only dynamic parts of the scene. It turns out that such assumptions hold for a large amount of televised footage.

Specifically, we uniformly sample parameteres in $\mathcal{P}$ from the following domain:

$$x_0 \in [x_c - 0.2W, x_c + 0.2W], \quad y_0 \in [y_c - 0.1H, y_c + 0.1H] \tag{5.7}$$

$$\theta_{shld} \in [0, \pi), \quad \theta_{elb} \in [0, 2\pi) \tag{5.8}$$

$$r \in [0.2, 1], \quad d \in D \tag{5.9}$$

, where (W, H) is dimension of the video frame with its center at (x_c, y_c), θ are joint angles of shoulders and elbows, and D is a small set of depth configurations from a frontal-viewpoint (e.g. arms are always in front of torsos).

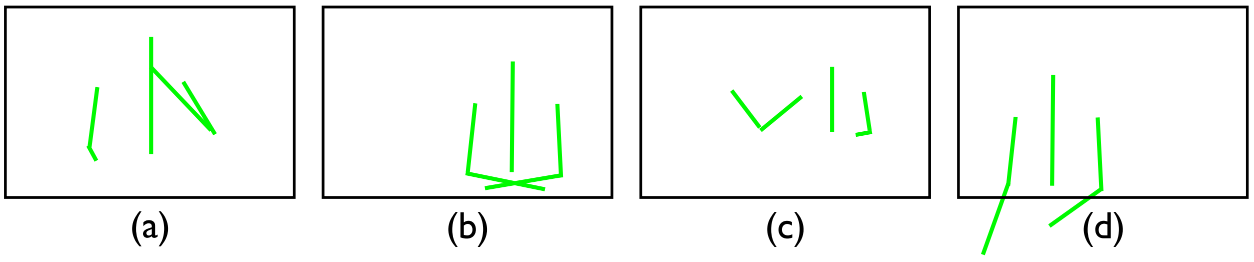


Figure 5.5: Pose synthesis. The pose pool consists of diverse poses in different scales and locations. It includes challenging cases, such as (from left to right) self-occlusion, interacting parts, various scales, and truncation.

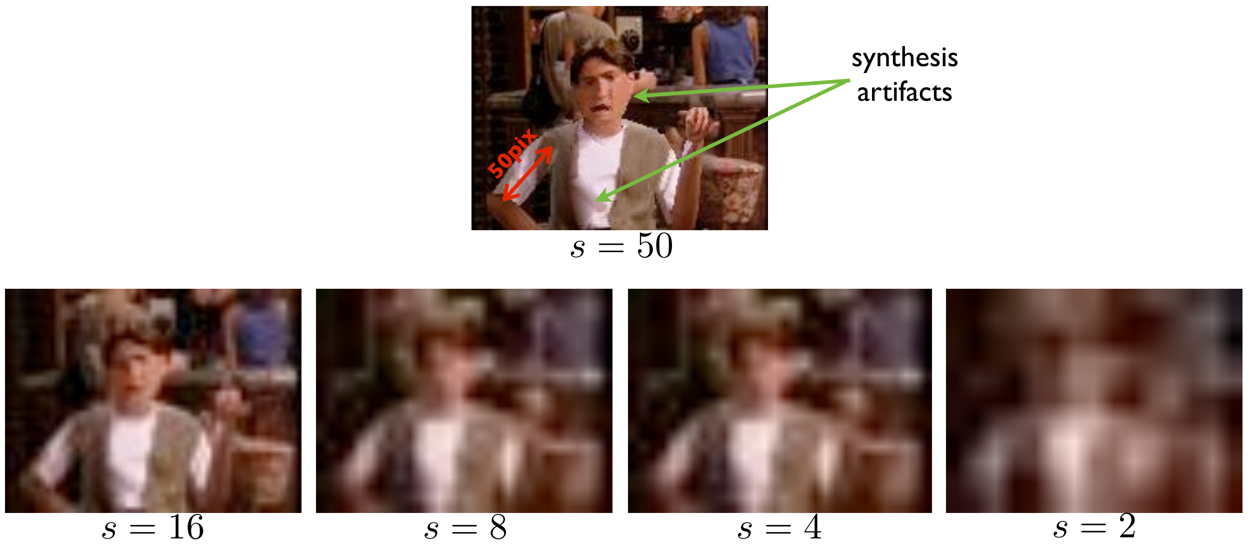


Figure 5.6: Low resolution color space. Typically, synthesized images are presented with significant artifacts (**top**). In order to mitigate the effect, we render images in low resolution space. Even with quite low resolution ($s = 4$), humans can reasonably estimate upper body poses. The resolution s is represented by the length of full upper arm in pixels.

5.3.3 Low resolution rendering

An ideal synthesis engine needs to generate a training set with two computationally demanding properties; (1) it needs to be *photorealistic* enough so that it matches well with the real test images, and (2) it needs to be *comprehensive* enough to cover all expected test frames. In order to enhance photorealism, one can add more parameters to the synthesis engine, e.g. shearing parameters for out-of-plane rotation or parameters for face expression and detailed clothing models. However, since each of those introduces another axis in the joint parameter space, the number of images one need to synthesize grows exponentially.

One way to address both problems is to use low resolution training images. As shown in Fig. 5.6, the artifacts from our simple warping methods and heuristics for resolving occlusion is significantly reduced in low resolution space. Interestingly, humans still reasonably perform pose estimation with a resolution as low as $s = 4$. We quantitatively show in Sec. 5.5 that our recognition system exhibits similar perception.

Furthermore, one needs to synthesize only a small set of images that are distinct in appearance in the low resolution space. Quantitatively, reducing resolution by a factor of R reduces the number poses by a factor of R^{2K}, where K is the number of parts. This means that one may use lower sampling rates in pose synthesis described in Sec. 5.3.2. We show by experiments in Sec. 5.5 how pose estimation accuracy correlates with the number of training images, given various resolutions.

As a by-product of using low-resolution training images, one can significantly speed up the synthesis process by directly rendering the images in the low-resolution space. This is achieved in Sec. 5.3.1 by projecting the 2.1D layered model and labeled/queried poses to smaller scales. In practice, synthesizing images of $s = 16$ in Fig.5.6 takes 0.04s, while $s = 50$ takes 0.28s in a 3.0GHz single-core desktop.

Synthesizing blur: Our layered synthesis engine produces crisp edges across layers, while actual low-resolution images are quite blurred. We mimic this blur during our synthesis by rendering at b times the target resolution, and then subsampling the rendered image with antialiasing. Such a procedure actually improves performance in two ways. First, generated image features appear more realistic. Second, we can now represent a larger family of poses, specifically b^{2K} more poses. Perhaps surprisingly, we show that one can still resolve such "sub-pixel" pose configurations in a low-resolution image space.

5.4 Inference

Intuitively, given highly customized training data for a particular video, training an accurate recognition machine may be greatly simplified. We verify this hypothesis by performing upper body pose estimation using very simple image features and learning/inference algorithms. As image features, we use (low-resolutional) raw pixel values of the entire frame in perceptually uniform color spaces such as LUV or LAB. As a classifier, we use a nearest-neighbor regressor. That is, for each test frame we independently find the training image with the least L_2 distance in given feature space, and report its pose (after converting to skeleton format).

$$(I^*, \mathcal{P}^*) = \underset{(I,P)}{\arg\min} \|\Phi(I) - \Phi(I_{test})\|_2 \tag{5.10}$$

It is widely accepted that the most crucial property of robust image features is their invariance to affine deformation, luminance, albedo, etc. As a result, modern image features are based on normalized edge-orientations ([13], [56]) or gabor-like filter responses ([74], [50]). However, in the scenario of tracking where there exist large consistency in appearance, *less* invariant features are likely to perform competitively. In Sec. 5.5, we compare pixel-value features with edge-based features to demonstrate this idea.

5.5 Experimental results

Dataset We use *Friends* dataset [76] to investigate the effect of key parameters (resolution, features, size of training data) and to compare our approach with state-of-the-arts. Of the 18 test clips in the dataset, we use only 13 containing frontal view of humans. These 13 clips are grouped and concatenated to form 5 longer videos, each of which contains a single

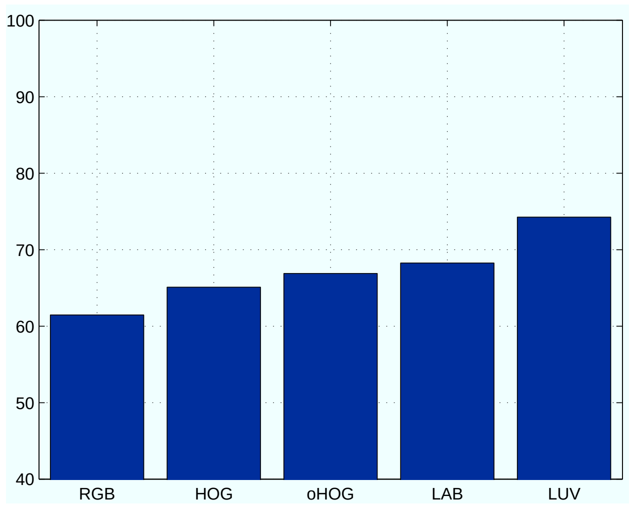
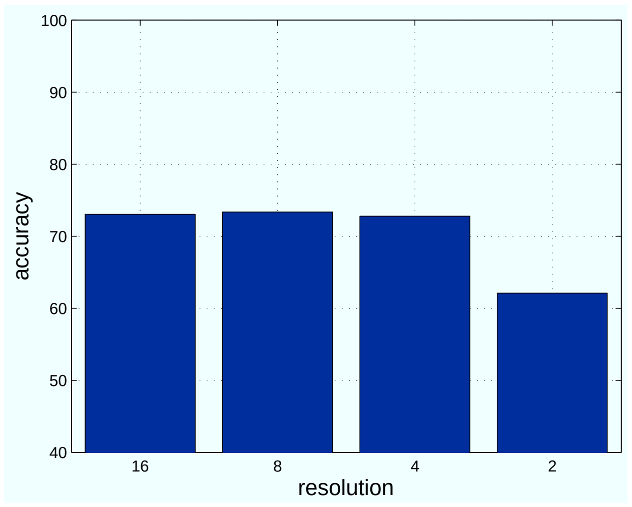


Figure 5.7: Image features. We show how the choice of image feature affects the pose estimation accuracy. Overall, with the exception of RGB, *less* invariant features performs better than standard image features. Particularly, raw pixel values in perceptually uniform color space such as LUV and LAB significantly outperform standard edge-based features (HOG). Color-augmented HOG (oHOG, [88]) with no contrast normalization performs better than HOG.

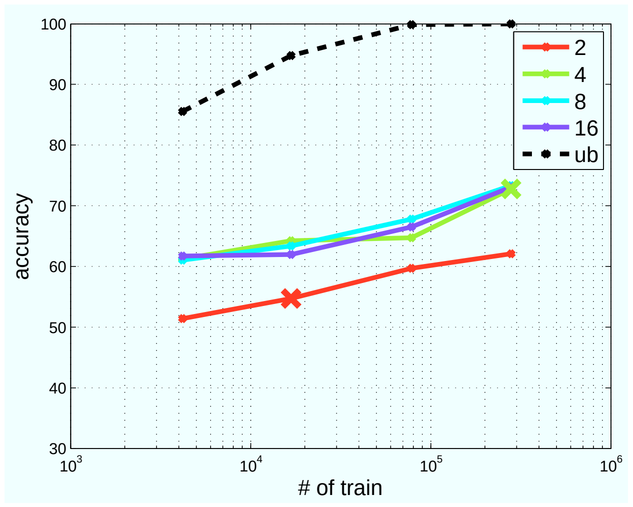
character and background scene. (5 *take*s are split into 13 clips in the dataset.) The length of videos range from 50 to 120 frames. Background scenes are mostly stable, but there exists mild motion due to movement of camera and/or objects. The target task is to accurately predict joint locations of arms (elbows and wrists), which is known to be notoriously hard compared with other body parts.

Evaluation In all diagnostic experiments, we use as a scalar evaluation metric, the percentage of correctly predicted joints with 25-pixel threshold in a normalized scale. This radius roughly corresponds to the width of fist of given character. We consider 4 joints; two elbows and wrists. When comparing with other approaches, we present the result with full range of thresholds as in [106] and [76].

We first compare features of various degree of invariance (Fig.5.7). **HOG** has rich machinery to generate invariant feature space, such as spatial and orientation pooling and contrast normalization [13]. We attempted to alleviate its invariance by building 3-channel color his-



(a)



(b)

Figure 5.8: Low resolution color features and training data. In **(a)**, we show the accuracy of pose estimation with respect to the resolution of features. The number of training images is fixed (∼280k). The x-values are scales represented by the length of full upper arm (See Fig. 5.6). In **(b)**, we plot performance as a function of the number of training images (i.e sampling rate in pose space) for each resolution of LUV features. "ub" denotes an upper bound obtained by reporting the training pose closest to the ground truth test pose, measured in high-resolution image coordinates. This plot reveals that high accuracy (85%) can be theoretically obtained with a small number of rendered training images (∼4k). The "**x**" denotes the number of unique quantized poses that are resolvable at a fixed resolution (only shown for $s = 2$ and $s = 4$). The additional accuracy one obtains for $s = 2$ reveals the benefit of rendering "subpixel" poses, as discussed in Sec. 5.3.3.

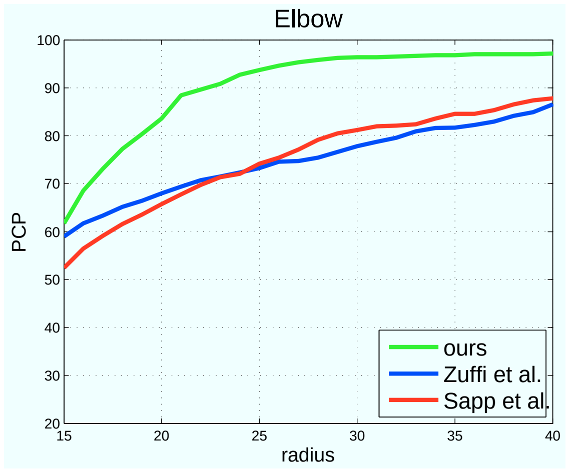
tograms (inspired by OpponentSift in [88]) and removing contrast normalization (**oHOG**). This modification yields 3% improvement in accuracy. We explored different sizes of spatial/orientation bin, and report the best one.

We also evaluate the simplest and the least invariant type of features, *pixel-value* features. Interestingly, these features work better than the best setting of HOG by significant margin (with the exception of **RGB** color space). We found that using perceptually uniform color space such as **LUV** or **LAB** is important, presumably because they were designed to make L_2 distance more meaningful.

The next question we answer is about the working resolution of color features and its interplay between the number of training frames. Interestingly, as shown in Fig.5.8a, we achieve competetive accuracy using quite low resolution ($s = 4$), and observe a sharp drop for $s = 2$. This is consistent with visual inspection of the pixel data as well; it is quite hard for a human to see structure at low resolutions (Fig.5.6).

In fact, the correlation between feature resolution and accuracy is more subtle, since the accuracy also depends on the number of rendered poses (or the pose space sampling rate mentioned in Sec.5.3.3). Intuitively, the number of visually distinguishable poses must decrease at low resolutions. This observation suggests that one may need to render only those poses with unique quantized configurations at a given resolution. Fig5.8b shows that "sub-pixel" pose configurations further improves accuracy. An upper-bound analysis reveals that a small number of poses ($\sim 4K$) can potentially achieve a quite high accuracy ($\sim 85\%$), but this may require complex image matching function (capable of deforming images while matching). Rather, our approach is to synthesize a set of deformations with consistent depth-layering.

Lastly, we compare our approaches with the state-of-the-art on the *Friends* dataset. [76] uses an ensemble of tree models, each of them rooted on one of the 6 parts and temporally



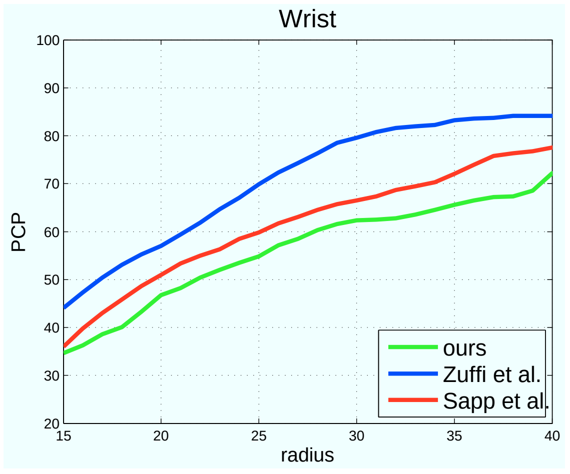
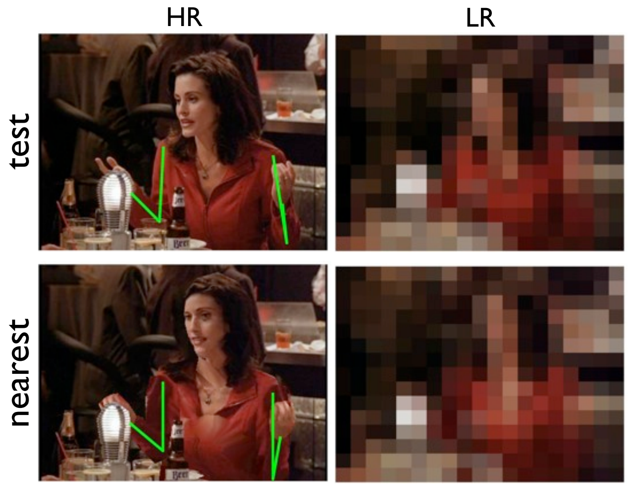


Figure 5.9: We compare our best result (LUV features, $s = 5.7$, $\sim$ 280k training frames) to other two state-of-the-arts reported on *Friends* dataset, [76] and [106]. We outperform with large margin (**25%** at 25-pixel threshold) in the task of predicting elbow locations (**left**). We perform competetively in predicting wrist locations (**right**)

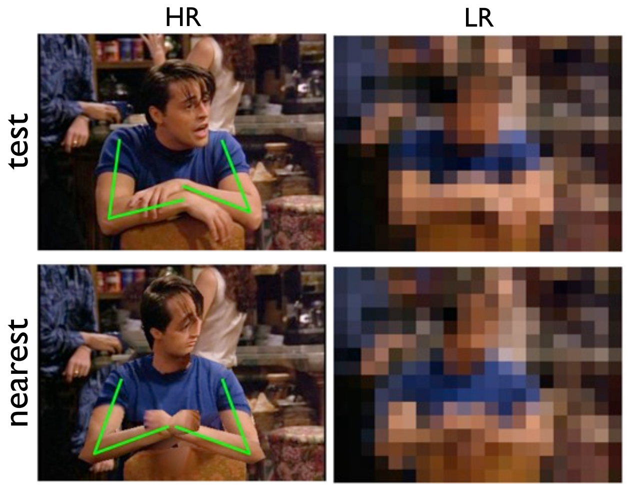
linked only through roots, to approximate underlying loopy spatiotemporal model. [106] uses optical flows and learned 2D articulated shape models as means to exploit pixel informations of adjacent frames and to propagate part assignments temporally. Both methods use optical flows and designated off-the-shelf hand detector based on assumptions on skin colors.

Our nearest-neighbor regressor predicts elbow locations significantly better than other two methods; 93.7% versus 73.2% and 74.2% at 25-pixel threshold. For wrist, our methods is less accurate than [76] and [106]; 54.8% versus 69.9% and 59.8% at 25-pixel threshold (Fig.5.9). Unlike other two methods, we independently estimate poses in each frame without using temporal models or motion features. Plus, there is no extra effort for detecting hands.

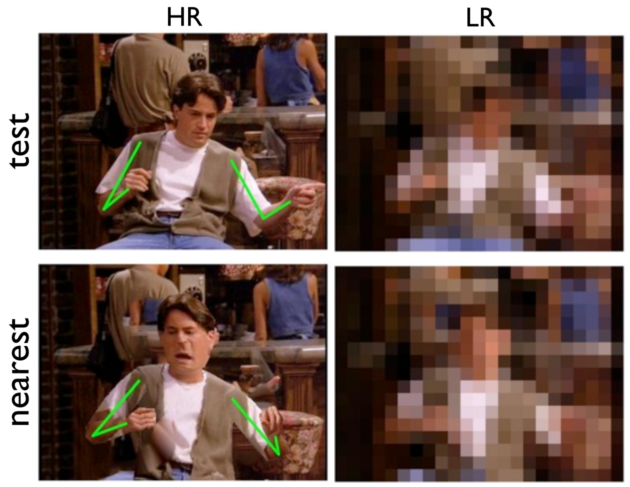
One of the benefits of simplicity in features and learning algorithm is that visualizing and understanding how the predictor works is straightforward (Fig.5.10). For instance, a common mistake is that hands are often confused by background objects with similar color (failure of *explain-away* model). In addition, our approach of using customized synthetic frames facilitate further error analysis. For instance, one can attempt to *copy* the test frame using the given synthesis engine to understand false positives.



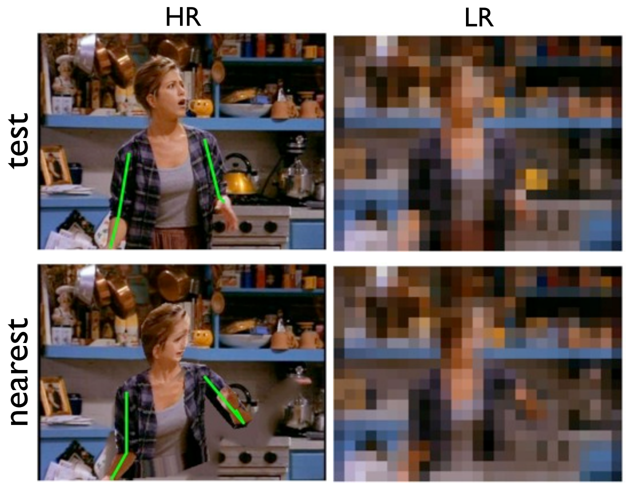
(a)



(b)



(c)



(d)

Figure 5.10: Pose estimation. We independently estimate upper body pose in each test frame by finding the nearest training frame in low-resolutional color spaces. Although simple, our method is robust against (self) occlusion **(a)** and challenging interaction of parts **(b)**. A common mistake is due to confusing color in the background **(c, d)**, which are also readily confused by human in such low-resolution space.

5.6 Conclusion

In this chapter, we described an approach of using synthetic training dataset to train models highly customized to the particular video. We show that, with simple image-based rendering algorithms, one can generate reasonably photorealistic training data that captures important modes of variation (human poses) of given video, while maintaining its invariants. We showed that this custom training data greatly simplify learning and inference. We demonstrated our approach on the challenging task of estimating upper body pose of humans in videos.

5.7 Summary of thesis

In this thesis, we discussed various methods for accurately tracking people and their poses. We considered two types of tracking task; 1) tracking bounding boxes and 2) tracking poses. For each of those tasks, we proposed methods for a) improving apperance models and b) exploiting temporal cues. In terms of tracking bounding boxes, we proposed multiresolution features and motion features that are useful in detecting small people. In terms of tracking poses, we focused on combinatorial inference on part models and highly tuned appearance models. We demonstrated our approaches using various datasets including public benchmarks such as Caltech Pedesrian Detection Benchmark, MindsEye dataset, and *Friends* dataset.

Bibliography

[1] `http://people.cs.uchicago.edu/~pff/latent`.

[2] Minds eye dataset. `http://www.visint.org/index.html`.

[3] C. Anderson, P. Burt, and G. Van Der Wal. Change detection and tracking using pyramid transform techniques. In *Proc. SPIE Conference on Intelligent Robots and Computer Vision*, pages 300–305, 1985.

[4] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, pages 1–8. IEEE, 2008.

[5] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Proc. CVPR*, volume 1, page 4, 2009.

[6] O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using hough transforms. In *CVPR*, pages 2233–2240. IEEE, 2010.

[7] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 24:1–24:11, New York, NY, USA, 2009. ACM.

[8] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910. IEEE, 2012.

[9] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE TPAMI*, 33(3):500–513, 2011.

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

[11] R. Collins et al. *A system for video surveillance and monitoring*, volume 102. Carnegie Mellon University, the Robotics Institute, 2000.

[12] N. Dalal. *Finding people in images and videos.* PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2006.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005.

[14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*. IEEE, 2005.

[15] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *ECCV*, pages 428–441, 2006.

[16] D. Demirdjian, L. Taycher, G. Shakhnarovich, K. Grauman, and T. Darrell. Avoiding the "streetlight effect": Tracking by exploring likelihood modes. *ICCV*, 1:357–364, 2005.

[17] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models of multi-class object layout. In *ICCV*, 2009.

[18] P. Dollar, B. Babenko, S. Belongie, P. Perona, and Z. Tu. Multiple component learning for object detection. *ECCV 2008*, pages 211–224, 2008.

[19] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. *BMVC*, 2010.

[20] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV VS-PETS*, 2005.

[21] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[22] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[23] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE TPAMI*, 34(4):743–761, 2012.

[24] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *ICCV03*. Citeseer, 2003.

[25] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, pages 726–733, 2003.

[26] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari. 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *International Journal of Computer Vision*, 99(2):190–214, 2012.

[27] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE PAMI*, 31:2179–2195, 2009.

[28] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. A mobile vision system for robust multi-person tracking. In *CVPR*, 2008.

[29] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[30] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop, 2007.

[31] A. Fathi and G. Mori. Human pose estimation using motion exemplars. In *ICCV*, pages 1–8, 2007.

[32] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[33] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *Computer Vision and Pattern Recognition, Anchorage, USA, June*, 2008.

[34] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE PAMI*, 99(1), 5555.

[35] R. Fergus, P. Perona, A. Zisserman, et al. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2. Citeseer, 2003.

[36] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, June 2008.

[37] D. Gavrila. Pedestrian detection from a moving vehicle. *ECCV*, pages 37–49, 2000.

[38] D. Gavrila and V. Philomin. Real-time object detection for smart vehicles. In *ICCV*, pages 87–93, 1999.

[39] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999.

[40] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International journal of computer vision*, 73:41–59, 2007.

[41] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE TPAMI*, 32(7):1239–1258, 2010.

[42] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge Univ. Press, 2000.

[43] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[44] L. Huang and D. Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics, 2005.

[45] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. Technical report, Institute of Mathematics of the Romanian Academy and University of Bonn, September 2012.

[46] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using region alignment. *IEEE TPAMI*, 1997.

[47] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003.

[48] M. Jones and D. Snow. Pedestrian detection using boosted features over many frames. In *ICPR*, 2008.

[49] S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 38–44. IEEE, 1996.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[51] I. Laptev and T. Lindeberg. Local descriptors for spatio-temporal recognition. *Spatial Coherence for Visual Motion Analysis*, pages 91–103, 2006.

[52] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, volume 2, pages II–97. IEEE, 2004.

[53] M. Lee and I. Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In *CVPR*, volume 2. IEEE, 2004.

[54] Z. Lin, G. Hua, and L. S. Davis. Multiple instance feature for robust part-based object detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 405–412, 2009.

[55] T. Lindeberg. *Scale-space theory in computer vision.* Springer, 1994.

[56] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[57] B. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[58] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel SVMs is efficient. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[59] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on pattern analysis and machine intelligence*, 14(7):710–732, 1992.

[60] J. M. McCarthy. *Introduction to theoretical kinematics.* MIT press, 1990.

[61] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[62] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE PAMI*, 23(4):349, 2001.

[63] D. Nilsson. An efficient algorithm for finding the M most probable configurationsin probabilistic expert systems. *Statistics and Computing*, 8(2):159–173, 1998.

[64] D. Nilsson and J. Goldberger. Sequentially finding the N-best list in hidden Markov models. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 17, pages 1280–1285. Citeseer, 2001.

[65] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *IEEE CVPR*, pages 193–199, 1997.

[66] D. Park and D. Ramanan. N-best maximal decoders for part models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2627–2634. IEEE, 2011.

[67] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. In *Computer Vision–ECCV 2010*, pages 241–254. Springer, 2010.

[68] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. *ECCV*, 2010.

[69] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár. Exploring weak stabilization for motion feature extraction. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2882–2889. IEEE, 2013.

[70] R. P. Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators.* Richard Paul, 1981.

[71] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics*, volume 4, pages 3099–3104. IEEE, 2004.

[72] L. Pishchulin, A. Jain, M. Andriluka, T. Thormahlen, and B. Schiele. Articulated people detection and pose estimation: Reshaping the future. In *CVPR*, pages 3178–3185. IEEE, 2012.

[73] D. Ramanan, D. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE PAMI*, pages 65–81, 2007.

[74] X. Ren and D. Ramanan. Histograms of sparse codes for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3246–3253. IEEE, 2013.

[75] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *Proc. CVPR*, pages 1–8, 2007.

[76] B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *CVPR*, 2011.

[77] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1. IEEE Computer Society; 1999, 2000.

[78] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. *International Journal of Computer Vision*, 2009.

[79] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *CVPR*, pages 750–757. IEEE, 2003.

[80] E. Shechtman and M. Irani. Space-time behavior based correlation. In *IEEE TPAMI*, 2007.

[81] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3D human figures using 2D image motion. *ECCV*, pages 702–718, 2000.

[82] L. Sigal and M. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *CVPR*, volume 2, pages 2041–2048. IEEE, 2006.

[83] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. 2013.

[84] F. Soong and E. Huang. A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition. In *icassp*, pages 705–708. IEEE, 1991.

[85] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE PAMI*, 28(9):1372–1384, 2006.

[86] Z. Tu and S. Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *IEEE PAMI*, pages 657–673, 2002.

[87] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE PAMI*, 30(10):1713–1727, 2008.

[88] K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1582–1596, Sept. 2010.

[89] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[90] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 2005.

[91] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *CVPR*, pages 1030–1037. IEEE, 2010.

[92] H. Wang, M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[93] J. Y. Wang and E. H. Adelson. Representing moving images with layers. *Image Processing, IEEE Transactions on*, 3(5):625–638, 1994.

[94] X. Wang, T. X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. *International Journal of Computer Vision*, 2009.

[95] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *IEEE TPAMI*, 2012.

[96] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[97] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2411–2418. IEEE, 2013.

[98] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li. Robust multi-resolution pedestrian detection in traffic scenes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3033–3040. IEEE, 2013.

[99] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered object models for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1731–1743, 2012.

[100] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. *Computer Vision and Pattern Recognition*, 2011.

[101] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, pages 1385–1392. IEEE, 2011.

[102] C. Yanover and Y. Weiss. Finding the M Most Probable Configurations Using Loopy Belief Propagation. In *NIPS*, page 289. The MIT Press, 2004.

[103] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[104] C. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[105] L. Zelnik-Manor and M. Irani. Event-based analysis of video. In *CVPR*, volume 2, pages II–123. IEEE, 2001.

[106] S. Zuffi, J. Romero, C. Schmid, and M. J. Black. Estimating human pose with flowing puppets. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3312–3319, 2013.