Towards Human Pose Estimation in Video Sequences

by

Georgii Oleinikov

B.Sc., V. N. Karazin Kharkiv National University, 2010
M.Sc., V. N. Karazin Kharkiv National University, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

The University Of British Columbia

(Vancouver)

January 2014

Abstract

Recent advancements in human pose estimation from single images have attracted wide scientific interest of the Computer Vision community to the problem domain. However, the problem of pose estimation from monocular video sequences is largely under-represented in the literature despite the wide range of its applications, such as action recognition and human-computer interaction. In this thesis we present two novel algorithms for video pose estimation that demonstrate how one could improve the performance of a state-of-the-art single-image articulated human detection algorithm on realistic video sequences. Furthermore, we release the UCF Sports Pose dataset, containing full-body pose annotations of people performing various actions in realistic videos, together with a novel pose evaluation metric that better reflects the performance of current state of the art. We also release the Video Pose Annotation tool, a highly customizable application that we used to construct the dataset. Finally, we introduce a task-based abstraction for human pose estimation, which selects the "best" algorithm for every specific instance based on a task description defined using an application programming interface covering the large volume of the human pose estimation domain.

Preface

The work on the contents of Chapter 6 was done in collaboration with Gregor Miller. In this section of the thesis we would like to highlight the parts of the research project, performed by Gregor Miller and the author. Gregor Miller's contributions towards the project are as follows:

- An idea of a task-based abstraction targeting non-expert users together with a task-to-algorithm mapping
- Formulation of the target description and organization of the condition matrix 6.1
- Part of the experiments for the evaluation of pose estimation algorithms

The author of this thesis made the following contributions to the project:

- Formulated the input type and output requirements that are included in the task description
- Surveyed the pose estimation literature to make sure the task description covers sufficiently large volume of the problem space
- Selected the algorithms that were included in the framework
- Manually annotated training and test images with description such as amount of clutter and occlusion
- Performed part of the experiments for the evaluation of pose estimation algorithms
- Analyzed the results of the experiments and determined the contents of the condition matrix 6.1
- Suggested the task-to-algorithm mapping procedure 2

Also, Kevin Woo helped us to create the UCF Sports Pose Dataset by making high-quality annotations using the Video Pose Annotation tool (Chapter 3).

Table of Contents

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Little for his invaluable support and encouragement throughout my Master's thesis, for multiple prolific and stimulating discussions, important suggestions and insights as well as for the constant involvement and the readiness to help. I want to thank Gregor Miller for the interesting and productive collaboration on the pose estimation abstraction, for multiple inspiring and motivating ideas and suggestions. I would also like to thank Prof. Woodham for thoroughly reading my thesis and providing many valuable comments.

I would like to thank Ankur Gupta for his help as well as many thoughtful discussions and insightful comments on my ideas throughout my Master's thesis. I want to thank Kevin Woo for making the high quality video annotations as well as his important feedback and suggestions regarding the Video Pose Estimation tool. I would like to thank Daria Bondareva for her invaluable continuous support throughout my thesis that helped me to overcome multiple challenges, for her readiness to help, and for the work she did on the video annotations.

Dedication

I dedicate this thesis to my Mom

Nina Oliinykova

who made it possible for me, without whom I would not be here. Everything that she gave me from my earliest childhood played a role in who I am now. The English lessons that she taught me when I was a small kid were exciting and inspiring, and I would never forget the verb flower garden that she drew for me. The Math classes that we went through together determined my future career path. And I will always remember the door with a picture of a smiling computer on it, behind which I first experienced the exciting world of personal computers.
For an amazing childhood, for all the love she gave to me I am infinitely grateful to my Mom.

Chapter 1

Introduction

Human pose estimation is an area in Computer Vision that aims to identify the correct pose of a person, often defined by the position of body joints and limbs, given an image, video sequence or depth data. Unconstrained, real-world human pose estimation is a challenging problem that has been widely studied in Computer Vision. It has great potential to assist a wide range of applications, such as indexing of images and videos, activity understanding and action recognition, automatic surveillance and human-computer interaction [37].

In this thesis we address the problem of 2D human pose estimation in monocular video sequences. In particular, we are interested in developing algorithms that determine the 2D coordinates of 14 body joints in every frame of a given video sequence. We assume that there is only one person in the recording and the full body is visible in all frames. Figure 1.1 demonstrates an example of the desired output of a video pose estimation algorithm. Although in this thesis we consider full-body poses only, the techniques developed in this work can be applied towards upper-body or other kinds of poses.

Furthermore, we also consider the pose estimation problem from the opposite perspective. Namely, how one should select the right pose estimation algorithm for a particular problem, and what semantic language would one use to describe it. We are seeking to develop an abstraction that would allow one to describe any pose estimation problem for which the solving algorithms exist in the literature or may potentially emerge in future. The problem description includes specification of the input data and requirements for the pose estimation results as well as relevant task conditions. In addition, we are interested in a system that would accept the problem description and return the results of a pose estimation method that suits the problem best.

1.1 Motivation

Most of the pose estimation applications in real-world scenarios provide one with video sequences. Video cameras and other sensors often work in real-

Figure 1.1: Pose estimation in video sequence example. The pose in each frame is represented by a body graph with more than 14 joints, with connections between joints denoting parts of limbs, torso and head.

time, producing streaming data. In fact, there are relatively few applications where pose estimation in images is required while video data is not available. Surprisingly, most of the current pose estimation algorithms focus on estimating the pose from single images. However, it seems that pose estimation over time is critical for improving the estimations [25], and the temporal information should be utilized for improved performance. This motivates us to develop two video pose estimation algorithms, attempting to fill in the gap between the wide range of applications of pose estimation in video and the lack of algorithms that utilize the full available data by focusing on video sequences. We develop these algorithms with an application to sports video analysis in mind.

In order to evaluate and train video pose estimation algorithms one would need the ground truth. To the best of our knowledge, there are no

publicly available datasets with unconstrained real-world video sequences and fully annotated human poses (see Section 2.1.3). We firmly believe that the shortage of data is one of the reasons for the lack of video pose estimation algorithms. Therefore, in this thesis we are also determined to create a video dataset with annotated full-body poses.

In our opinion, the lack of data is mainly due to the fact that it is time-consuming and generally difficult to annotate all frames of a video sequence with human poses. It is much harder than to annotate bounding boxes or provide action labeling, since various pose representations usually consist of 10 to 20 body joints, each of which must be manually adjusted for every frame. Furthermore, providing one dataset with fully annotated poses would not solve the problem of the shortage of data, simply because it cannot fit all the potential requirements in the degree of data complexity and type. Certain algorithms may require high-quality data with annotated upper bodies, while other would need full-body annotated videos with heavy occlusions and action labelings. Therefore, we decide to take a thorough approach and create a tool, that would allow one to annotate skeletal poses of humans in video sequences.

The challenge to develop accurate pose estimation algorithms is not the only obstacles that prevent them from being effectively embraced in real-world applications. For most software engineers who are not experts in Computer Vision it is challenging to utilize the best pose estimation algorithm for their needs. With the state of the art advancing fast it is hard to track down the best algorithm for every specific case of input conditions and output requirements. Furthermore, it is hard for non-experts to implement these algorithms and keep them up-to-date with the state of the art. In order to address these problems we develop an abstraction over human pose estimation together with a task-to-algorithm mapping, which selects the best pose estimation algorithm based on task description.

1.2 Outline

Data Preparation. In order to help solve the data shortage problem we introduce a Video Pose Annotation tool, which allows one to annotate skeletal poses people in video sequences. The skeletal representations may have various forms, enabling the tool to be adjusted for the specific needs of the user, such as annotation of upper bodies or hands only. The tool features a simple yet powerful graphic user interface, and the annotation process is assisted by automatic detection, tracking and interpolation. Furthermore, we

introduce the UCF Sports Pose dataset, which is a UCF Sports dataset with annotated full-body poses. It features more than 150 sports video sequences with high-quality full-body annotations made with the aforementioned annotation tool. In addition, we propose a new pose estimation evaluation metric that in our opinion better reflects the performance of the common state-of-the-art algorithms for pose estimation. To the best of our knowledge, until most recently both the tool and the dataset were the only ones of their kind[1].

Pose Estimation. Most of the recent single-image pose estimation algorithms build on top of the pictorial structures model [2]. It represents the human body as a tractable tree graph, making full search possible via dynamic programming. However, the direct application of the same technique to video sequences is not possible in practice. The single-frame tree graph turns into a loopy intractable model, where inference complexity would grow exponentially with the number of frames.

The most natural way to overcome the above is to make an algorithm that works on top of the detections returned by a single-image pose estimator. It would not search through all possible combination of poses, but it may give a good approximation, depending on the quality of the single-image pose detector. Wang *et al.* [51] uses dynamic programming to search through multiple single-image detections of a state-of-the-art pose estimator, utilizing color information to score pose configurations. Zuffi *et al.* [58] build their work on the idea of using the optical flow to integrate image evidence from multiple frames. They iteratively propagate best single-image detections from every frame to the adjacent ones and then refine and shrink the set of poses for every frame.

The first video pose estimation method that we introduce in this thesis builds on the two ideas described above[2]. We first run a state-of-the-art single-image pose detector on every frame of a video sequence, selecting several best detections. Then we propagate the poses in every frame to neighbouring frames in order to expand the set of poses for every frame. Finally, we select the best-overall combination of poses throughout the obtained sets of poses. We use Flexible Mixture of Parts (FMP) [55] as a state-of-the-art single-image pose estimation algorithm. We refer to this method as the shortest path approach, since it involves minimizing the cost of transferring from a pose in the first frame to a pose in the last frame.

[1] See Section 2.1.3.

[2] Note that current algorithm was developed independently from [51] and [58]. See Section 4.4 for comparison of these approaches.

One of the alternatives to the above is to look at some of the ways to deal with the temporal dimension from the action recognition approaches literature. Most of the common methods utilizing local features involve either computation of optical flow or trajectories [52] [54] or utilize local volumetric space-time features such as SIFT3D [45] or HOG3D [28]. Fragkiadaki *et al.* [19] use optical flow to segment body parts and propagate segmentations over time. Tian *et al.* [49] extend the Deformable Parts Model [14] to the temporal dimension by replacing the 2D HOG filters [9] with their volumetric versions HOG3D and use them for action detection.

The second video pose estimation method that we develop resorts to a different approach. Instead of using 3D spatio-temporal features or optical flow only, we look at how the 2D features change over time along the paths of the optical flow, thus combining the appearance and flow information in a single framework. Furthermore, in contrast to the approaches above we are able to do inference in the current frame and several previous frames at once. The latter is possible because we relax the connections between all joints in every frame but the current one, only leaving the temporal connections between joints and their instances in the past. The resulting structure is a tree, which enables full search via dynamic programming (see Figure 5.1). The relaxation of the body edges in previous frames does not have as much impact as one may think, because we set their expected positions in accordance with the backward optical flow around the body joints in the current frame. The idea of conversion of the intractable model spanning more than one frame to a tree structure is the most similar to the work by Sapp *et al.* [44], who decompose the model into an ensemble of several tree-structured models that cover the edge relationships of the full model. We refer to our method as the detection approach, since it estimates the pose in every current frame independently, taking into account several previous frames.

Abstraction. In this thesis we also develop an abstraction over human pose estimation together with a task-to-algorithm mapping. The abstraction features an interface, allowing the user to describe the pose estimation problem, which includes input conditions and output requirements. The interface is flexible enough to describe most of the possible variations in a problem description. The task to algorithm mapping encompasses expert knowledge about performance of several pose estimation methods and uses it to select the best one according to the given problem definition. We design this system as a part of OpenVL, an abstraction over Computer Vision, which currently encompasses tasks such as segmentation, image registration, correspondence, detection and tracking.

To summarize, in this thesis we make the following five key contributions:

- Video Pose Annotation tool, which allows one to annotate skeletal poses of humans in video sequences
- UCF Sports Pose dataset, containing realistic videos with full-body annotations, together with a new pose estimation evaluation metric PCP2D
- Video pose estimation method, demonstrating a way to improve a pose estimation algorithm for video sequences
- Novel video pose estimation method, embracing both temporal and appearance information in a single framework
- An abstraction over human pose estimation together with a task-to-algorithm mapping, which selects the best algorithm according to the given problem description.

1.3 Organization

This thesis is organized as follows. In Chapter 2 we discuss related work as well as briefly give necessary background on algorithms that are essential for understanding this thesis. We present the Video Pose Annotation tool and the UCF Sports Pose dataset in Chapter 3. Afterwards we focus on pose estimation algorithms for video sequences and introduce the shortest path approach in Chapter 4. Then we proceed to the detection approach for pose estimation in Chapter 5. We describe the abstraction over pose estimation in Chapter 6 and finally finish with conclusions in Chapter 7.

Chapter 2

Related Work

In this chapter we survey the relevant literature in Section 2.1 and then give necessary background required for understanding of the material in this thesis in Section 2.2.

2.1 Literature Overview

In this section we first survey related work on pose estimation algorithms for various forms of input data and output results from the task-result perspective, which is be needed in Chapter 6 in order to justify the design of the abstraction interface and the selection of the algorithms in the framework. Then we proceed to the overview of the existing methods targeting 2D human pose estimation in monocular videos. We further describe relevant datasets and annotation tools, and then proceed to existing abstractions over Computer Vision and briefly describe OpenVL.

2.1.1 Pose Estimation Algorithms

3D human pose estimation is a hard problem that has been researched most successfully in the setting of depth images. A method for super-realtime estimation of 3D positions of body joints and pixel-wise body-part labelings based on randomized decision forests was introduced by Shotton *et al.*in [46], which was a technology behind the initial release of Kinect^{TM}. Fanelli *et al.* [13] tackled a problem of real-time head pose estimation from depth data using random regression forests.

Another class of methods considered the problem of 3D human pose estimation using sources of data other than depth images. Yu *et al.* [56] introduced a method for monocular 3D pose estimation from video sequences using action detection on top of 2D deformable part models. Amin *et al.* [1] presented a method for 3D pose estimation from multiple calibrated cameras, incorporating evidence from every camera obtained with 2D pictorial structures. The problem of determining 3D shape of the human body together with its pose was considered by Guan *et al.* [20]. Although estimating

a 3D pose solely from a 2D image is an under-constrained problem, it has been tackled by Simo-Serra *et al.* [48] by jointly solving 2D detection and 3D inference problems.

The problem of 2D body pose estimation has traditionally been approached with variations of pictorial structures framework [2]. Recently, Yang and Ramanan introduced a flexible mixture of parts model [55], which extended the deformable parts model [14] for articulated 2D human detection with considerable improvement to the state of the art. The state of the art was further improved among others by Rothrock *et al.*, who used a compositional and-or graph grammar model together with segmentation [43]. Also, Kinect-style body-part labelings were obtained by Ladicky *et al.* [31], combining part-based and pixel-based approaches in a single optimization framework. Hara and Chellappa introduced a super-realtime 2D pose estimator with the help of multidimensional output regressors along the body part dependency paths [21]. The problem of head and face orientation estimation from images was tackled by Maji *et al.* [34] and Zhu and Ramanan [57].

It is easy to see that with such an abundance of algorithms performing various tasks the development of an abstraction to select the best algorithm for every specific case would be beneficial.

2.1.2 Pose Estimation in Video

There is a large literature on 2D human pose estimation in single images. Similarly, many methods were devoted to the pose tracking problem, which often assumes correct manual initialization in at least one of the frames of the video sequence. However, general 2D pose estimation in monocular video sequences is largely underrepresented in the literature.

Nevertheless, several recent papers focused on pose estimation in video without any requirement for supervision. Some papers exploit the idea of relying on confident detections. Ramanan *et al.* [41] require that the video sequence contains an easily detectable canonical pose. They find the pose with an accurate canonical pose detector and use it for instance-specific appearance training, which is subsequently utilized to find poses in all frames independently. Buehler *et al.* [6] use a similar approach by identifying keyframes with reliable detections and filling in the intermediate frames taking into account temporal consistency. Ferrari *et al.* [16] first reduce the search space by highlighting the foreground with segmentation applied on top of the results of a human detection algorithm. Then they do single-frame pose detections and refine them with a spatio-temporal instance-specific model trained on

reliable detections. Wang *et al.* [51] searches for a best-overall combination of poses obtained from a single-image pose detector, taking into account temporal and appearance coherence.

Other methods use optical flow to exploit coherence of the information from consecutive frames. Fragkiadaki *et al.* [19] use segmented body parts and propagate segmentations over time with the help of optical flow. Zuffi *et al.* [58] exploit optical flow to propagate best single-image detections to the adjacent frames and refine and shrink the poses for every frame in an iterative process.

Simultaneous inference over more than one frame presents a challenge to deal with loopy intractable models, which necessitates approximate inference as in [53]. Alternatively, one may attempt to convert the intractable model into one where exact inference is possible. Sapp *et al.* [44] decompose the loopy model into an ensemble of several tree-structured models that cover the edge relationships of the full model.

2.1.3 Datasets

There exist a variety of single-image datasets with annotated poses [40] [10] [12] [11]. However, there are few video pose annotated datasets mostly due to the difficulties in manual annotation.

HumanEva [47] is a motion capture dataset providing both motion capture and video data of 4 subjects performing a set of 6 actions two times each. However, the environment is not realistic and the videos have static backgrounds, well centered persons and high contrast clothing, while the set of actions is limited. VideoPose 2.0 [44] is a video dataset with annotated arm joints every other frame. The dataset consists of 44 short clips, 2-3 seconds in length each, 1,286 video frames in total. Most recently, Jhuang *et al.*released J-HMDB [25], a video dataset with annotated full-body joint positions and human silhouettes derived from joints. It contains 21 action classes, 36-55 clips per action class 15-40 frames each, 31,838 video frames in total. Together with the dataset Jhuang *et al.*announce an annotation tool[1] that helped them to build J-HMDB. Its current web demo allows one to drag joints over the body and propagate annotations to the next frame.

2.1.4 Abstractions over Computer Vision

The idea of developing an abstraction for Computer Vision tasks is not new, and there have been numerous attempts towards it. Matsuyama and Hwang

[1] `http://files.is.tue.mpg.de/hjhuang/pose_annotation/html/avalidator.html`

introduced SIGMA [35], an expert system performing detection based on a learned appearance model, which is selected based on geometric context-dependent reasoning. Kohl and Mundy developed the Image Understanding Environment, an abstraction providing high-level access to vision methods, although requiring the understanding of all the algorithms underneath it [29]. Firschein and Strat introduced RADIUS [18], which helped the user choose best image processing algorithms based on geometric models, defined by the user. Konstantinides and Rasure developed a visual programming language in *Khoros*, which allowed its users to create vision applications by connecting components in a data flow [30]. However, it also required a thorough understanding of the vision algorithms, as the components it included were relatively low-level, featuring color conversions, spatial filtering and feature extraction.

More recently declarative programming languages such as ShapeLogic[2] and FVision [39] were introduced, which provided functionality as small low-level units, requiring expert knowledge about vision methods. Chiu and Raskar introduced Vision on Tap, a web-based tool featuring a high-level abstraction targeted for web developers [8], although its usage is limited due to its web interface.

Several openly available libraries, such as OpenCV [3], FastCV[3], OpenTL [38] and the Vision Toolbox[4], provide common Computer Vision functionality. These frameworks provide direct access to specific vision components and algorithms, but the context of usage and tuning of the parameters is essential, which requires expert Computer Vision knowledge.

Most recently Miller and Fels introduced OpenVL [36], an abstraction targeting a variety of Computer Vision problems from the task perspective. Currently, OpenVL is working with segmentation, correspondence and registration, while certain steps have been made towards tracking and detection. Human pose estimation can be considered as an articulated human detection problem and thus fits well into the OpenVL paradigm, which allows us to extend OpenVL with pose estimation.

[2] `http://www.shapelogic.org`

[3] `http://developer.qualcomm.com/mobile-development/mobile-technologies/computer-vision-fastcv`

[4] `http://www.mathworks.com/products/computer-vision`

2.2 Relevant Algorithms

In this section we go over some algorithms that are essential for understanding this thesis. We start with a brief description of the Flexible Mixture of Parts model (FMP) [55]. We further go over dynamic programming and the Viterbi algorithm and then proceed to the distance transform of sampled functions [15]. Also, we cover the basics of optical flow and explain the notation of median optical flow which we use as a tracking algorithm.

2.2.1 Flexible Mixture of Parts

Our work largely builds on top of FMP and we briefly describe it in this section. It is a human pose estimation method from single images, based on a mixture of non-oriented pictorial structures.

Model. The model is a tree graph (V, E) covering the human body, where each node i is located at pixel $p_i = (x_i, y_i)$ and is assigned a filter type f_i. Every filter type f_i is associated with a particular HOG filter [9] representing a specific mode of the appearance of part i. Every body part has several appearance modes, which cover the most common cases of the part's appearance. The score of a configuration of body part positions $p = \{p_i\}_{i=1}^{K}$ and part types $f = \{f_i\}_{i=1}^{K}$ in an image I is defined as follows:

$$\begin{aligned} S(I, p, f) = &\sum_{i \in V} b_i^{f_i} + \sum_{(i,j) \in E} b_{ij}^{f_i f_j} + \\ &\sum_{i \in V} \omega_i^{f_i} \cdot \phi(I, p_i) + \sum_{(i,j) \in E} \omega_{ij}^{f_i f_j} \cdot \psi(p_i - p_j), \end{aligned} \tag{2.1}$$

where $\psi(d_x, d_y) = [d_x^2 \quad d_y^2 \quad d_x \quad d_y]^T$ is a deformation spring model and $\phi(I, p_i)$ is an image feature vector extracted at location p_i. The first two terms of (2.1) represent the appearance compatibility score, the third term defines the appearance score, while the last term is a quadratic-cost deformation score. Note that, in practice, in order to reduce computation during inference the assumption on $\omega_{ij}^{f_i f_j}$ is relaxed, stating that deformation spring models depend only on the filter type of child i:

$$\omega_{ij}^{f_i f_j} = \omega_{ij}^{f_i}. \tag{2.2}$$

Inference. The inference procedure corresponds to maximizing (2.1) with respect to p and f. This can be done efficiently using dynamic programming (see Section 2.2.2) with message passing of the form

$$score_i(f_i, p) = b_i^{f_i} + \omega_i^{f_i} \cdot \phi(I, p_i) + \sum_{j \in kids(i)} m_j(f_i, p), \tag{2.3}$$

$$m_j(f_i, p) = \max_{f_j} \left[b_{ji}^{f_j f_i} + \max_{p_j} \left(score_j(f_j, p_j) + \omega_{ji}^{f_j f_i} \cdot \psi(p_j - p_i) \right) \right]. \tag{2.4}$$

The computational cost of (2.4) for each body part is $O(L^2H^2)$, where L is the total number of body part pixel locations and H is the number of HOG filters per part. This can be reduced to $O(LH^2)$ with the help of the distance transform [15], described in Section 2.2.3. Assumption (2.2) reduces the cost further to $O(LH)$.

Learning. The supervised learning paradigm with negative $\{\hat{I}^{(n)}\}_{n \in N}$ and positive labeled $\{(I^{(n)}, p^{(n)}, f^{(n)})\}_{n \in P}$ training examples is employed. The scoring function is linear in model parameters $\beta = (b, \omega)$ and can be rewritten as $S(I, z) = \beta \cdot \Phi(I, z)$, where $z^{(n)} = (p^{(n)}, f^{(n)})$. Therefore, the model is learned in the form

$$\begin{aligned} &\arg\min_{\omega, \epsilon_n \geq 0} \frac{1}{2} \beta \cdot \beta + C \sum_n \epsilon_n, \\ \text{s.t.} \quad &\forall n \in P \quad \beta \cdot \Phi(I^{(n)}, z^{(n)}) \geq 1 - \epsilon_n, \\ &\forall n \in N, \forall z \quad \beta \cdot \Phi(I^{(n)}, z) \leq -1 + \epsilon_n. \end{aligned} \tag{2.5}$$

The latter is a quadratic programming problem, which can be optimized with an out of the box solver such as the cutting plane solver in [17] or stochastic gradient descent in [14].

2.2.2 Dynamic Programming

We extensively use dynamic programming throughout this thesis. In this section we discuss relevant dynamic programming ideas.

The Viterbi algorithm. Suppose we are given a sequence of variables $X = (X_1, \ldots, X_n)$, each of which can take one of the m values $\{s_j\}_{j=1}^k$. Furthermore, there are scores $S_i(X_i)$ associated with the choice of particular assignment of values to variables X_i as well as scores $S_{i-1,i}(X_{i-1}, X_i)$ for particular co-assignments of the values in adjacent variables X_{i-1}, X_i. The Viterbi algorithm solves the following problem:

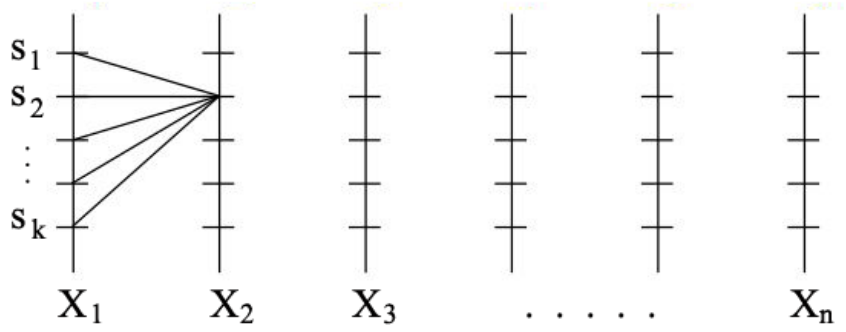


Figure 2.1: The Viterbi algorithm. For every value of X_2 the best value of X_1 is computed, then the process continues up to X_n.

$$\arg\max_X S(X), \tag{2.6}$$

$$S(X) = \sum_{i=1}^{n} S_i(X_i) + \sum_{i=2}^{n} S_{i-1,i}(X_{i-1}, X_i) \tag{2.7}$$

In order to do this, for every X_i starting with X_2 we can compute the best candidate value for X_{i-1} (see Figure 2.1):

$$score_1(X_1) = S_1(X_1), \tag{2.8}$$

$$score_i(X_i) = S_i(X_i) + \max_{X_{i-1}} m_{i-1}(X_{i-1}, X_i), \quad i > 1, \tag{2.9}$$

$$m_{i-1}(X_{i-1}, X_i) = (score_{i-1}(X_{i-1}) + S_{i-1,i}(X_{i-1}, X_i)), \tag{2.10}$$

$$ind_i(X_i) = \arg\max_{X_{i-1}} m_{i-1}(X_{i-1}, X_i), \quad i > 1. \tag{2.11}$$

Here $score_i(X_i)$ stores the total accumulated score (2.7) up to X_i, and $ind_i(X_i)$ is the index of the best assignment to X_{i-1} for every assignment of X_i. This process is generally referred to as *message passing* from X_i to X_{i+1}.

After finishing the message passing procedure $score_n(X_n)$ would contain the final configuration scores. The best combination of assignments (2.6) can be then obtained by taking the maximum value of $score_n(X_n)$ and applying

backtracking, a process of consecutively recovering the best assignment of X_i:

$$X = \bigotimes_{i=1}^{n} (ind_{i+1} \circ \cdots \circ ind_n \circ id)(\arg\max_{X_n} score_n(X_n)), \tag{2.12}$$

where $\otimes$ denotes Cartesian product, $\circ$ denotes function composition, id is the identity function.

Tree structures. The Viterbi algorithm can also be applied to other cases when the graph connecting variables X_i forms a tree. The whole inference procedure stays the same, with the only difference that message passing (2.9) has to take into account all children $kids(i)$ of node i:

$$score_i(X_i) = S_i(X_i) + \sum_{j \in kids(i)} \max_{X_j} m_j(X_j, X_i). \tag{2.13}$$

2.2.3 Distance Transform of Sampled Functions

In this section we describe the distance transform of sampled functions [15], as its understanding is essential in Chapter 5.

One dimension. Let $G = \{g_1, \ldots, g_n\}$ be a one-dimensional grid. The goal is to compute $D_f(p), E_f(p)$ for every p in a grid $H = \{h_1, \ldots, h_m\}$:

$$D_f(p) = \min_{q \in G} P_2(p, q), \tag{2.14}$$

$$E_f(p) = \arg\min_{q \in G} P_2(p, q), \tag{2.15}$$

$$P_2(p, q) = a(p - q)^2 + b(p - q) + f(q). \tag{2.16}$$

This can be done via full search in $O(nm)$ time. The distance transform however is able to compute this in O(n+m) in the following way. The first step is the computation of the lower envelope of parabolas $a(p-q)^2 + b(p-q) + f(q)$. This can be done in linear time by using simple algebra. During the second step the values of $D_f(p)$ are filled in for all p in grid H by selecting the appropriate parabolas in the lower envelope (see figure 2.2).

Two dimensions. Let $G = \{g_1^1 \ldots g_n^1\} \times \{g_1^1 \ldots g_1^k\}$ be a two-dimensional grid with an arbitrary function $f : G \to \mathbb{R}$ defined on it. We are aiming to compute $D_f(x, y)$ for every (x, y) in a grid $H = \{h_1^1 \ldots h_m^1\} \times \{h_1^1 \ldots h_1^l\}$:

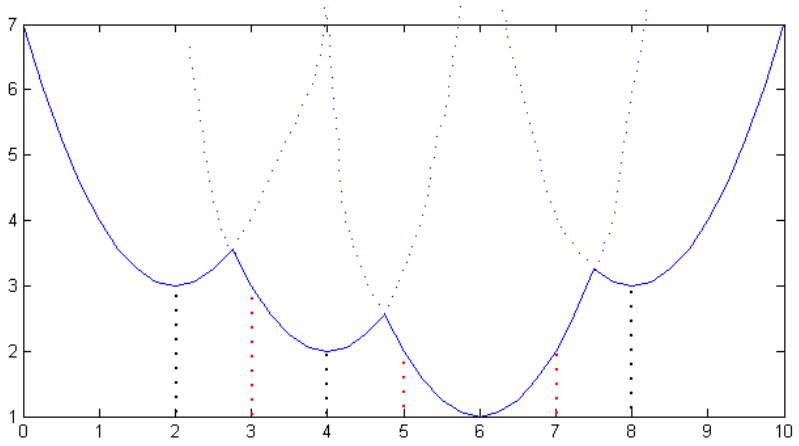


Figure 2.2: Lower envelope of parabolas for the distance transform. In this example $b = 0$ and parabolas are centered at points in grid $G = \{2, 4, 6, 8\}$. Blue contour corresponds to lower envelope of parabolas, while dotted parts of the parabolas represent their parts that do not constitute a part of it. Dotted red vertical lines correspond to grid $H = \{3, 5, 7\}$, in which the values of $D_f(p)$ will be filled. In this example $E_f(3) = 4, E_f(5) = 6, E_f(7) = 6$.

$$D_f(x, y) = \min_{(x',y')\in G} P_2((x, y), (x', y')), \tag{2.17}$$

$$P_2((x, y), (x', y')) = a_x(x - x')^2 + b_x(x - x') + a_y(y - y')^2 + b_y(y - y') + f(x', y'). \tag{2.18}$$

Since the first two terms in (2.18) do not depend on y, the equation above can be rewritten as

$$D_f(x, y) = \min_{x'}[a_x(x - x')^2 + b_x(x - x') + D_f|_{x'}(y)]. \tag{2.19}$$

Thus, we can use one-dimensional distance transform along the y axes and then use it again along the x axis on the result.

2.2.4 Optical Flow

In this thesis we frequently resort to optical flow. By optical flow we mean a class of methods attempting to calculate the motion between two consec-

utive video frames taken at times t and $t + \Delta t$ for small values of Δt. The brightness constancy equation is often utilized:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t), \tag{2.20}$$

where $I(x, y, t)$ is the brightness of the pixel (x, y) at time t. The equation above stays that brightness of pixels, potentially belonging to the same object in the video sequence should stay the same. Optical flow algorithms usually encompass additional assumptions and constraints to improve the optical flow accuracy. They can be roughly divided into local and global approaches. Local methods such as Lucas-Kanade [33] are often more robust to noise, while global methods such as Horn-Schunck [22] produce a dense flow field.

In this paper we use an optical flow algorithm that combines local and global approaches, attempting to yield a dense flow field that is robust to noise. It uses the brightness constancy assumption, the gradient constancy assumption and a discontinuity-preserving spatio-temporal smoothness constraint. It is based on [4] and [5], and we use its Matlab implementation by Liu [32]. However, the choice of particular optical flow algorithm is not important for us, and it can be replaced with any other method.

We also use the notion of median optical flow throughout this thesis. By median optical flow in an image area we mean the median values of flow coordinates Δx and Δy in the image region. It can be used as a simple tracking algorithm.

Chapter 3

Data Preparation

There are multiple applications of human pose estimation in video sequences, such as human-computer interaction, entertainment, surveillance and sports video analysis. Surprisingly, there are very few methods that focus on 2D pose estimation in video in comparison to a large number of single-image algorithms being published every year (see Section 2.1.2). We strongly believe that one of the main reasons for that is a lack of video datasets with annotated poses (see Section 2.1.3), on which these algorithms could be trained and/or evaluated. The availability of such datasets would immediately benefit the research community working on pose estimation and may potentially attract more research into this field.

While there are many video datasets with annotated people locations (for tracking) and actions (for action recognition), there are few fully annotated realistic pose video datasets, and untill very recently none of them included full-body annotations (see Section 2.1.3). We think this is mainly due to the fact that it is very hard and time consuming to annotate poses for all frames of a video sequence. For instance, in contrast to annotating a bounding box or an action label, for a full-body pose annotation of a 3-second video sequence one would have to provide locations of 14 body joints for every of the 3×30 frames, resulting in more than 1000 mouse clicks. Furthermore, annotation of a pose requires much more precision than annotation of a box, and it is difficult to make the annotations consistent throughout the video sequence. Evidently, without any annotation tool the whole annotation process becomes impractical.

In this chapter we make two key contributions, motivated by the arguments above:

- Introduce the Video Pose Annotation tool, allowing one to make fast and easy pose annotations in video sequences, featuring a user-friendly graphical interface and flexible design.
- Introduce the UCF Sports Pose dataset, consisting of full-body annotations for the UCF Sports Action dataset [42]. The annotations were produced with the above annotation tool.

To the best of our knowledge, until most recently the Video Pose Annotation tool was the only application aiding the task of manual annotation of poses in images or video sequences. Likewise, the UCF Sports Pose was the only video dataset providing full-body pose annotations in realistic environments. Recently Jhuang *et al.*released their J-HMDB dataset [25] together with an annotation tool. We compare it with our tool in Section 3.3.

This chapter is organized as follows. We introduce the Video Pose Annotation tool in Section 3.1 and then describe the annotated dataset that we obtain with its help in Section 3.2.

3.1 Video Pose Annotation Tool

The Video Pose Annotation tool enables fast and accurate annotation of human poses in video sequences, featuring finely tuned Graphic User Interface (GUI). The annotation process is aided by automatic pose initialization, tracking and per-joint interpolation.

The annotations that we use are defined by the 2D locations of all body joints that represent the structure of the body. The model of the human pose as well as the way it is aligned with the image data depends on the required level of detail. We follow Ramanan [40] and use the 14-joint skeleton body model defining the positions of arms, legs, hips, shoulders, neck and head. See Figure 3.1 for some examples of annotations that we expect for our model. Note that since we are interested in poses in videos, the ideal annotation would contain smooth movement of every joint throughout the sequence, preserving the length of body parts and their placements relative to the body.

3.1.1 Application Features

When working on a video sequence the user has the following options:

- Automatically estimate the pose in the current frame using a pose detector
- Automatically translate the current pose to any other frame using tracking
- Manually adjust the joints of the current pose

Automatic pose initialization is helpful because it often puts a number of joints at their desired locations. However, the current version of the

Figure 3.1: Several examples of 14-joint pose annotations. Pink and cyan lines cover right and left hands of a person, red and blue lines cover right and left leg correspondingly. Note that in contrast to Ramanan [40] we mark in red the actual right leg of the person, as opposed to the leg that is the most left in the image, assuming that the person always looks in the direction of the camera. This also makes difference when the limbs alternate, e.g. when the person is running sideways (best viewed in color).

pose detector often misses body parts or places them inaccurately. Usually manual adjustment is used afterwards to correct the pose. Furthermore, our current pose detector returns independent poses for every frame, and they often have slight differences in placement of head/shoulders/hips. As a result, detections in consecutive frames may have quite different poses, resulting in a very jittery annotation of poses overall. In order to increase the accuracy of annotations the tool utilizes tracking. Our experience reveals that tracking of a correct pose to the next frame produces substantially more accurate result than independent estimation of the pose.

Despite the good annotation results provided with workflow based on pose detections and tracking, it does have its disadvantages. The main

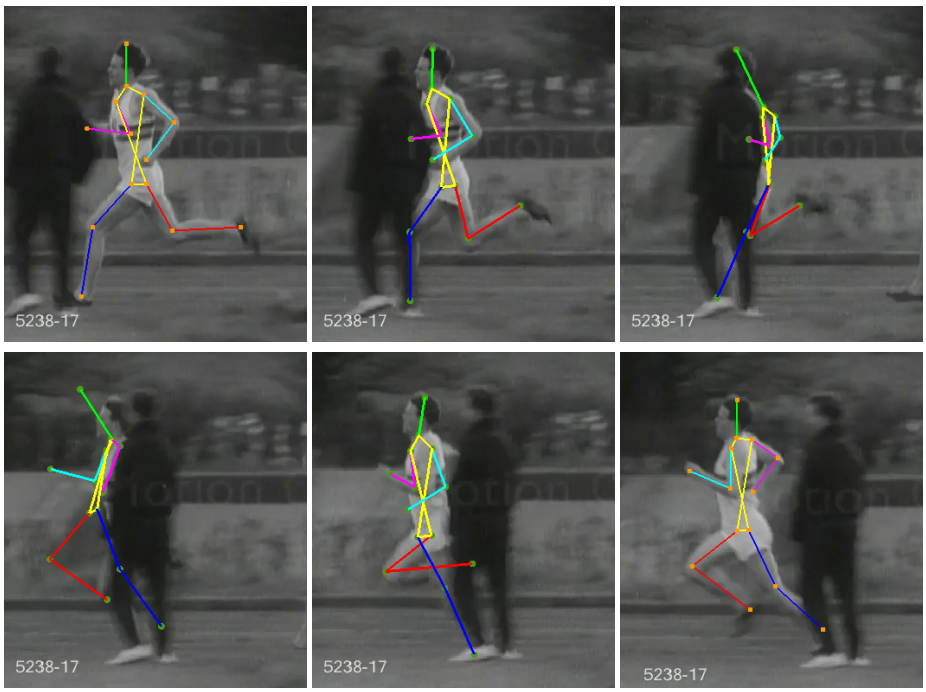


Figure 3.2: Annotation using tracking in a cluttered scene. The correct pose in a frame before occlusion was tracked forward, and the pose from a frame after occlusion was tracked back. It is very hard to guess the correct positions of occluded joints and maintain the right motion pattern.

downside is that one has to repeat the whole process for every frame, potentially adjusting every joint, which takes a lot of time. Furthermore, whenever occlusions or self-occlusions take place, it becomes very hard to correctly identify the positions of the missing parts in all frames, maintaining the right motion pattern (see Figure 3.2). In addition, we found that hard-to-notice subtle differences in consecutive frames may result in large displacements overall. For instance, the width of the hips in a video sequence may be changing all the time. It is hard to control such long-term deviations because one would have to go through all frames and separately adjust the incorrectly positioned joints.

In order to overcome the above difficulties interpolation between annotated poses is essential. We use a notion of a keypoint, which extends the common understanding of a keyframe. Every joint in every frame is either

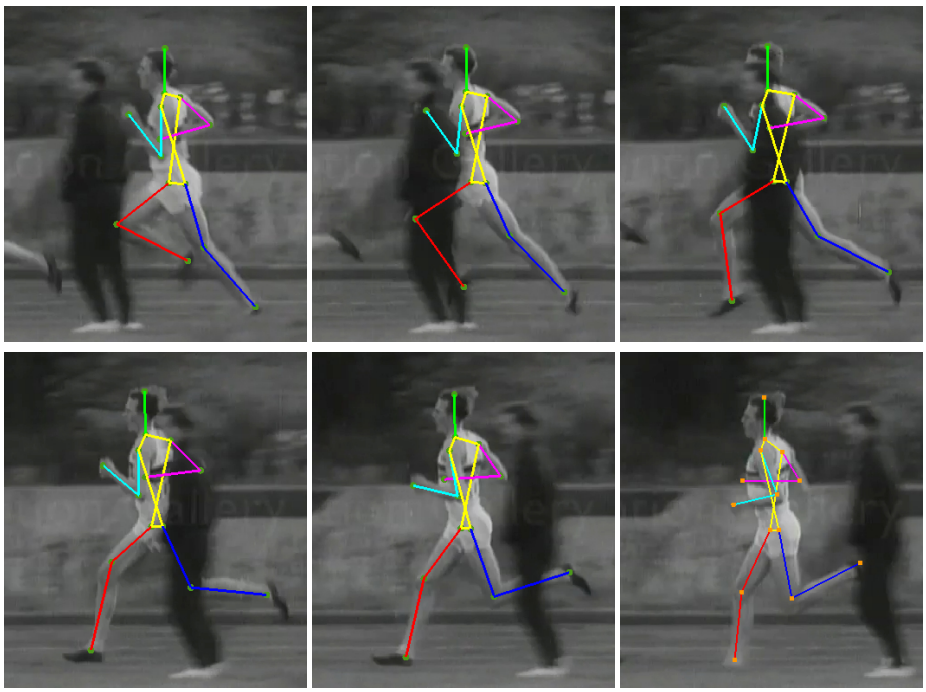


Figure 3.3: Annotation using interpolation in a cluttered scene. The interpolation on per-joint bases successfully resolves the occlusion problem. It allows one to specify only certain positions of a joint when it is visible, while all other positions get their values automatically.

marked as a keypoint or regular joint. The position of every regular joint is linearly interpolated in time between the closest left and right keypoints, and is adjusted accordingly when the position of any of the two keypoints changes. Every regular joint becomes a keypoint whenever it is manually adjusted or modified with detection or tracking. The user also has an option to remove any keypoint, making it a regular joint.

The latter interpolation procedure helps to solve the problems stated above. Instead of automatically estimating or tracking every pose to the next frame, one can do this every 5 or 10 frames, and interpolation would take care of the annotations in between. This not only saves time adjusting most of the joints in every frame, but also helps to recover unstable hips/shoulders and most importantly deal with occlusions (see Figure 3.3). Because of the complexity of human motion we find it particularly important that the

interpolation is done on per-joint bases. If one would resort to keyframes instead of keypoints, one would soon find out that most of the frames have at least one manually modified joint, which would turn every frame to a keyframe, and no interpolation would be performed.

We experiment with linear interpolation in two ways. In the first one, the joint position is linearly interpolated in image coordinates between its positions (x_1, y_1) and (x_2, y_2). While this interpolation keeps hips, shoulders and head more stable, we found out that it does not work very well on joints that cover hands, elbows, feet and knees mostly because human motion often produces swings that follow round trajectories. For example, feet in Figure 3.3 rotate relatively to knees, while knees rotate relatively to hips. Therefore, we use interpolation in polar coordinates for limb joints from ρ_1, ϕ_1 to ρ_2, ϕ_2, where ρ_1, ρ_2 are the distances from joint to its parent in the first and last frame of interpolation and ϕ_1, ϕ_2 are the angles relative to parent. This helps us to reduce the number of manual adjustments of joints. Note that the interpolation procedure described here could be replaced with any other algorithm, e.g. incorporate human motion models [53].

3.1.2 Graphical User Interface

The Video Annotation Tool was developed in a continuous usage-feedback-improvement loop. As a result we were able to develop a powerful yet simple GUI that suits the user's needs the best.

The main application window is shown in Figure 3.4. It consists of the image area, navigation bar, input/output panels and annotation and miscellaneous panel. Almost all of the functionality of the tool is hotkeyed, so that frequently repeated actions can be performed fast. All the changes made to the interface, such as last loaded video sequence or states of the check boxes are saved in the configuration file and loaded during subsequent runs.

Input/Output. The input panel determines the input video sequence, which could be loaded either by selecting a video file or an image sequence in a Load dialog window or by entering the path in the edit box. The output panel specifies the output `.mat` file, containing resulting annotations. Whenever a video sequence is loaded, the tool loads the corresponding annotations if they are found.

Annotation. The main functionality of the tool is gathered in the annotation panel. The `Detect` button performs automatic estimation of the pose in the current frame, while `Detect Fast` does local pose search based on the position and speed of the person in previous frames for the purpose

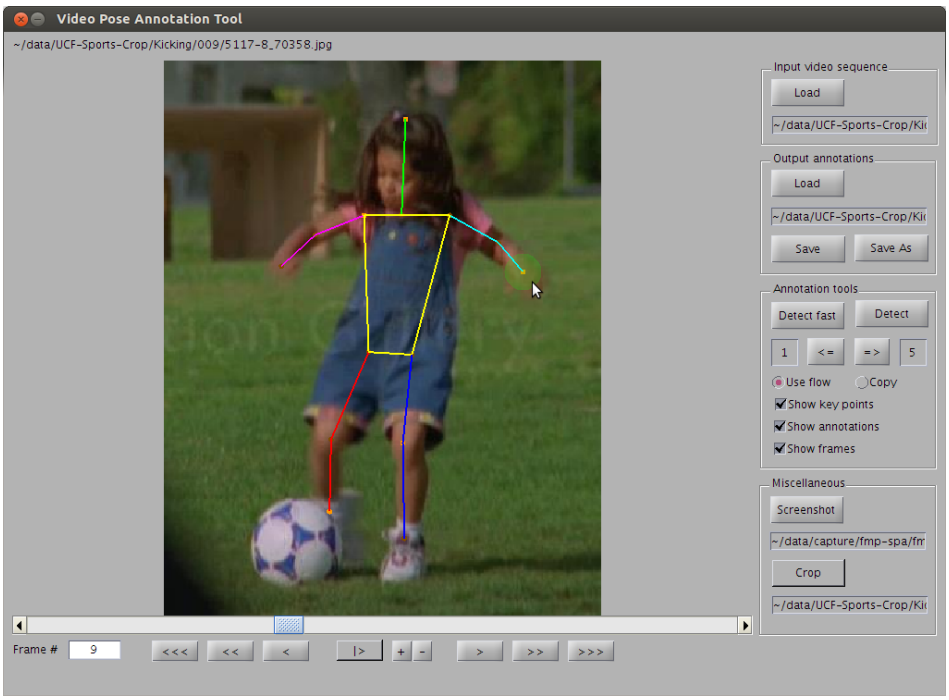
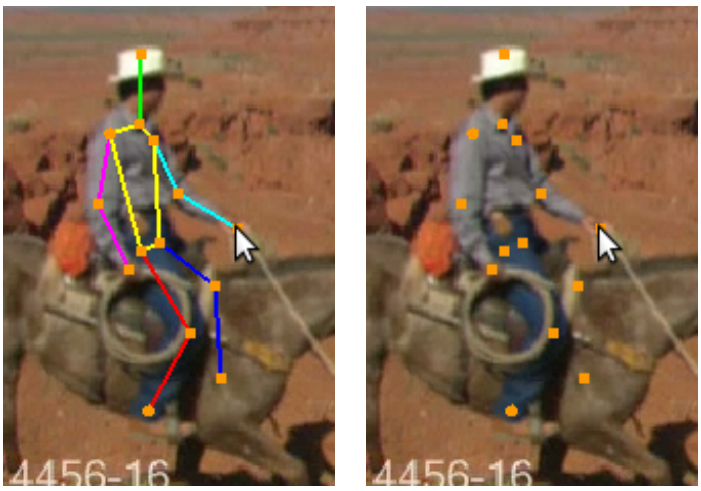


Figure 3.4: A screenshot of the Video Pose Annotation tool GUI. Annotations are displayed on top of the images as a colored stickman figure. Brighter colors for body joints represent keypoints, while darker correspond to regular joints. Hovering mouse over a joint pops up a transparent circle, identifying which joint is going to be affected. The left mouse button allows one to drag joints, while the right mouse button is used to remove the keypoint from the highlighted joint.

of reducing the computation time. The arrow buttons `<=` and `=>` perform tracking of the current pose back and forward correspondingly, while numbers in boxes nearby specify how many frames the pose should be tracked. The `copy` radio button enables the direct copy functionality, which may come in handy if tracking fails.

View and Navigation. The navigation bar allows one to browse the video frames back and forth, jump to a frame by number, play the video sequence with adjustable speed, etc. The corresponding annotations are displayed on top of the video frames in the image area. The user may manually adjust the annotations by dragging the joints around the image using the left



(a) (b)

Figure 3.5: Video Pose Annotation tool GUI functionality example. (a) It is hard to see where the limbs of the person are, while dragging the joints. (b) When annotations are hidden, only joints are shown when dragging, and lines do not occlude the limbs.

mouse button. Holding `Shift` results in groups of the joints being moved together, which is helpful when dragging the whole arm/leg/body together. Clicking the right mouse button releases the keypoint associated with the selected joint. Keypoints are highlighted with brighter color compared to regular joints, which can be turned off by unchecking the `Show keypoints` check box. Furthermore, one may want to uncheck `Show annotations` in order to hide the annotated pose. We found this useful when annotating videos of low quality/high motion noise, when it gets particularly hard to see what the right pose of the human is, with the annotations displayed on top (see Figure 3.5). Also, unchecking `Show frames` hides the images in case one wants to see how realistic the resulting motion of a stickman is. Finally, it is possible to take snapshots of the current image area with the help of `Screenshot` button.

3.1.3 Design

One of the objectives of the Video Pose Annotation tool is to be flexible enough to be applied in various scenarios. The dataset described in Section

3.2 includes full-body pose annotations, which might be useful for sports analysis applications. However, in other domains different pose representations might be required, such as upper body or hands only. In order for this tool to encompass potential changes in the body pose, we designed it with the principles of Object-Oriented Programming and flexibility in mind.

Figure 3.6 demonstrates the class hierarchy of the part of the application, responsible for body pose representation. `AbstractSkeleton` is the base class for all body part representations. `Skeleton2D` is the base abstract class for all "stickman" representations, which consist of 2D joint locations, sizes and connections between them. The distinction between the two is made in order to embrace potential classes that have information beyond the standard 2D information, such as 3D orientation or depth.

If one wants to annotate 2D poses with a different skeleton structure, they have to inherit the `Skeleton2D` class and provide implementation for methods representing the body graph: `skeletonSize, getParentIndexes, getPartConnections, getConnectionColors, getJointColors` and `getDragAdjacentJoints`. Also, one may modify the `SkeletonFactory` class, which creates the appropriate instance of the `AbstractSkeleton` class based on the number of joints.

We provide implementations for four body pose classes. `FullBody` represents a 14-joint body skeleton, `MidpointFullBody` expands the latter pose with joints in the middle of each limb and two additional joints on each side of the torso, resulting in a 26-joint body structure. Likewise, `UpperBody` is a 10-joint skeleton covering the upper body and `MidpointUpperBody` is its 18-joint expanded version. An example of a MidpointFullBody skeleton can be seen in Figure 1.1.

Every instance of a subclass of `AbstractSkeleton` also has a `createFrom` method, which serves the role of a constructor accepting instances of other classes inherited from `AbstractSkeleton`. This enables conversions between different pose classes, which may come handy since many pose representations share the same body parts. For instance, it is possible to convert `MidpointFullBody` to `FullBody` and back, `FullBody` can be converted to `UpperBody` etc.

We make two assumptions regarding the classes inherited from the base class `AbstractSkeleton`. First, the graph representing the body structure must be connected. Second, the number of joints in the graph should be different for every subclass of `AbstractSkeleton`. However, the application provides an easy way to overcome the assumptions above. If one wants to define a disjoint skeleton such as two arms, one can connect two disjoint subgraphs with an edge E in order to obtain a tree model and then specify

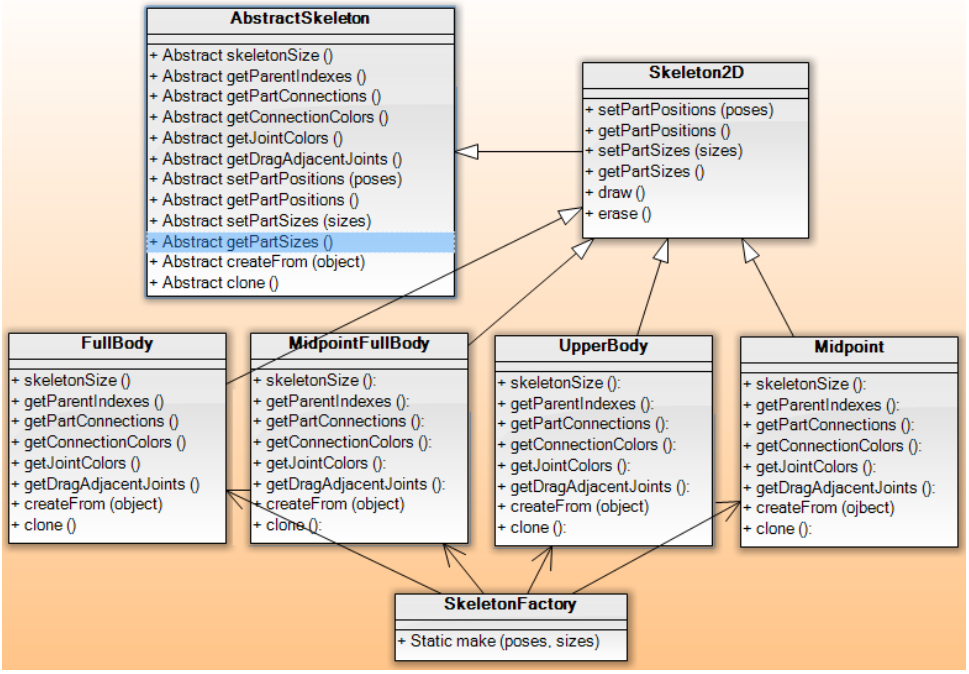


Figure 3.6: The UML class diagram for skeletal body models. `FullBody`, `MidpointFullBody`, `UpperBody`, `MidpointUpperBody` extend the `Skeleton2D` class, which extends the most abstract `AbstractSkeleton` class. `SkeletonFactory` is used to create appropriate `AbstractSkeleton` objects based on the joint information.

the color for the edge E to be transparent. Furthermore, if it happens that two different classes have the same number of joints, one may want to modify the `SkeletonFactory` class by introducing one more optional parameter, further distinguishing the classes between each other.

The current tool was designed such that every video sequence accepts only one annotation, thus not foreseeing simultaneous annotations of several people in one frame. A simple workaround for multiple-person sequences is to make a separate annotation file for every person in the video. However, the simultaneous annotations of multiple persons is made possible by the design of the application. This could be enabled by defining a single graph covering several skeleton models in the way described above and providing a multiple-person detection algorithm for initialization.

3.1.4 Implementation

We implemented the tool with Matlab and tested it on version R2011b. We use a Matlab implementation of Flexible Mixture of Parts [55] as a state-of-the-art pose detector (see Section 2.2.1) and median optical flow (see Section 2.2.4) as a tracking algorithm, based on Liu's Matlab optical flow implementation [32]. However, the detection and tracking algorithms can be easily changed based on the user's need. Such replacements may be necessary when changing the skeletal representation of the body.

3.2 Dataset

The research of this thesis was done mostly with applications to sports video analysis in mind. Therefore, we are most interested in datasets containing full body annotations in unconstrained real-world videos. Furthermore, action labeling might be potentially useful for the applications of pose estimation to action recognition.

The UCF Sports Action dataset [42] fits the description above and thus suits our needs. It contains more than 150 video sequences falling in one of the 9 action classes: diving, golf-swinging, kicking, lifting, riding-horse, running, skating, swinging and walking. The actions were collected from various sport recordings, typically featured on broadcast TV channels. Most sequences contain one or more people performing similar action.

In this work we release annotations for human poses in selected video sequences of the UCF Sports Action dataset. We limited ourselves to the following 7 action classes due to time constraints: golf-swinging, kicking, lifting, riding-horse, running, skating and walking. The people in these videos are roughly upright, which is in line with a some existing image datasets with annotated poses [16] [40] [12]. If a video sequence contains more than one person, we create annotation files for each one of them if they perform the action of their action class and are sufficiently unoccluded. We used the Video Pose Annotation tool (Section 3.1) to create these annotations. See some examples of the annotations in Figure 3.9. By releasing the dataset together with the annotation tool we hope to encourage more research into human pose estimation in video sequences and to lessen the gap between the abundance of its real-world applications and the lack of targeted algorithms.

3.2.1 Evaluation Metric

In this section we consider the question of the definition of the correct pose. We show that authors of different datasets and algorithms understand it differently and propose our own definition that in our opinion better suits the current state-of-the-art algorithms. We follow the tradition for the datasets in defining evaluation metrics for the consistency of the results, and propose a PCP2D evaluation metric that we suggest be used when reporting results on our dataset.

The most common evaluation metric for human pose estimation is the percentage of correct parts (PCP), reflecting the number of body parts estimated withing a certain distance threshold to their ground truth positions. Body parts are usually defined by the edges in the body graph. The most widely used version of PCP labels a body part as correct if the average distance of its joints to their ground truth positions is less than a threshold, which is defined by a fraction of the size of the ground truth body part [16]. The stricter version of PCP used by Ramanan [40] requires that both joints are within a threshold distance to their ground truth locations. The two versions of the PCP measure are the consequence of an ambiguous verbal definition of PCP by Ferrari [16]. In order to avoid such confusions in future we think it is important to address the question of how to define what is ground truth, which has not been addressed in the literature yet.

Let us consider the problem of pose estimation as the task of fitting a color skeleton in the image. The skeleton's right leg is red, left leg is blue, right arm is pink, left arm is cyan, torso is yellow and head is green. Johnson and Everingham [26] provide annotations for the Leeds dataset, such that the skeleton position always corresponds to the actual position of a person in the image (Figure 3.7 (a)). However, Ramanan [40] always fits the skeleton in the image so that the skeleton's red leg and pink arm are roughly on the left from its blue leg and cyan arm (Figure 3.7 (b)). Such labeling may be beneficial for various pose estimation algorithms such as FMP [55], because it allows them to build different appearance models for right and left parts of the body, which improves the pose estimation performance.

From the examples above one may see that the authors of different papers understand the notion of a correct pose differently. Johnson and Everingham require that a pose estimation algorithm evaluated on their dataset is capable of telling which side of the body is left and which is right. Ramanan, to the contrary, requires an algorithm to label everything on the left as red/pink, and on the right as blue/cyan. Therefore, an algorithm giving perfect results on the Leeds dataset would often confuse the limbs and

(a) (b)

Figure 3.7: Annotation examples from two different datasets. (a) Leeds dataset [26]. (b) People datset [26] (best viewed in color).

give lower performance on the People dataset and, conversely, the perfect algorithm for the People dataset will often fail on the Leeds dataset. Furthermore, such discrepancies become even more important when one deals with video sequences, where the relative horizontal placements of body parts change in one video sequence (Figure 3.8).

In order to address the above issues, we suggest two evaluation schemes. The first scheme requires an algorithm to be able to distinguish the left and right sides of a body, and uses standard PCP (either strict or loose) for evaluation. This scheme could be applied to the datasets that themselves distinguish the left and right sides such as Leeds dataset. The second scheme does not require an algorithm to have any knowledge about which side is which and allows it to freely confuse the left/right body parts. Although the first scheme describes the image best, we argue that the state of the art for 2D pose estimation is not able to differentiate the actual right and left body parts, and is not intended for this purpose [55]. Therefore, we introduce a modified version of PCP that we entitle PCP2D in order to elaborate the second scheme.

PCP2D is a metric for evaluating the percentage of correct body parts, allowing an algorithm to switch the left and right body parts. It operates on pairs of larger body parts that are defined by several edges in a body graph, which represent the arms, legs and two sides of the torso. There are two possible assignments of every pair of the left (L) and right (R) ground truth body parts to the left (L) and right (R) instances in the detection: $L \rightarrow L, R \rightarrow R$ and $L \rightarrow R, R \rightarrow L$. We compute the standard PCP (either

Figure 3.8: Example of video pose annotation with alternating limbs. Left/right positioning of left/right leg changes with time (best viewed in color).

strict or loose) for every assignment and take the highest one. The final PCP2D measure incorporates the highest PCP from every pair and also includes non-paired body parts such as the head.

We suggest using the PCP2D evaluation metric on the UCF Sports Pose dataset for algorithms that do not distinguish between the left and right sides of the body, as we think that using the standard PCP measure in such cases does not reflect the actual performance of an algorithm. However, standard PCP is an option for methods that are able to predict the left/right side labeling.

3.3 Discussion

Recently Jhuang *et al.*released an annotation tool, aiding manual annotation of human poses in video sequences [25]. In this section we briefly contrast it to the tool introduced in this section. The advantages of JHuang's tool in comparison to our application are as follows:

- Annotations come together with a direction-specific human silhouette
- The tool has web interface, which does not require any proprietary software such as Matlab

It is worth noting that the tool allows one to chose the silhouette based on direction, however its shape is defined by a set of joints and cannot be modified. Therefore, one may reconstruct similar silhouettes from annotations made with our tool. The drawbacks of the Jhuang's application according to its online demo are the following:

- Although the tool propagates the body poses to the next frame using optical flow, it does not have any interpolation functionality, and the pose must be adjusted for every frame
- Without interpolation there is no easy way one could deal with occlusions and self-occlusions when using the tool
- There is no easy way one could make temporally smooth annotations. The tool also does not support the video playback functionality, which would allow one to check the temporal consistency of annotations
- In contrast to our highly configurable application Jhuang's tool supports only one type of annotation which is the pre-defined full-body model
- At the time of writing the tool only features a web demo for a specific video sequences, and cannot be used for the user's data, doesn't support saving and loading annotations. Our tool in contrast can be used to browse annotations in a convenient way
- The source code of the tool is hidden behind the web interface, and thus cannot be modified. Our tool is available together with the source code and features flexible design for easy changes in algorithms underneath it

At the time of writing we have no access to J-HMDB annotations [25] and cannot directly compare the quality of the data. However, given the above considerations we believe that our annotations are more accurate and smooth, even when the exact location of body joints is unknown due to the occlusions.

Figure 3.9: Examples of the UCF Sports Pose dataset.

Chapter 4

Pose Estimation in Video: a Shortest Path Approach

In this chapter we focus on human pose estimation in video sequences. In particular, we are aiming to improve the state-of-the-art human pose estimation method in single images entitled Flexible Mixture of Parts (FMP) [55]. The key observations that motivate the work of this chapter are as follows:

- The FMP pose estimations are very noisy, often giving substantially different results in consecutive frames of a video sequence, even when the subject is almost static
- Often the pose estimation with the highest score obtained with FMP is not the best one, and there are better estimates among the top-scoring candidates
- Frequently the best estimation of pose is not present in the set of results, returned by FMP, while a similar pose is present among the best results in the adjacent frames. This happens mostly due to the double-counting problem, when two body parts cover the same image region

The above observations imply that it is possible to combine FMP with motion information to obtain better estimations of pose. The main contribution of this chapter is a method for human pose estimation in video sequences, improving the state-of-the art for pose estimation in single images.

This chapter is organized as follows. In Section 4.1 we describe the model, in Section 4.2 we explain the inference method. We discuss results in Section 4.3.

4.1 Model

The main idea of this method is to collect the best n outputs of FMP for every frame and expand it with additional examples that were missed by FMP using tracking, then find the best combination of poses throughout the whole video sequence with respect to a certain measure. We assume an offline setup, when the whole video sequence is given at once. The measure that we use when computing the best set of poses is the combination of local and pairwise scores. The local score of a pose in an image determines how well the pose matches the image, while the pairwise score between two consecutive pair of poses measure how well the poses are aligned with each other. We refer to this method as the shortest path approach.

Suppose we are given a sequence of video frames $I = \{I_t\}_{t=1}^{T}$. Let $p^t = \{p_i^t\}_{i=1}^{K}$ denote body pose in frame I_t, where $p_i^t = (x_i^t, y_i^t)$ is the pixel location of body part i, and $p = (p^1, \dots p^T)$ denote the total spatial configuration of body parts in T frames. Our goal is to find the best combination of poses throughout the T video frames:

$$p = \arg\max_{p \in P} S(I, p), \tag{4.1}$$

$$S(I, p) = \sum_{t=1}^{T} S_{loc}(I_t, p^t) + \sum_{t=2}^{T} S_{pair}(I_{t-1}, I_t, p^{t-1}, p^t). \tag{4.2}$$

Features. Although it is possible to use additional information such as color for the computation of local scores, we use only HOG features [9] in order to make the comparison to FMP fair. For the computation of pairwise scores we resort to tracking methods capturing motion information. Namely, we use optical flow (see Section 2.2.4).

Tracking. In the current model we use tracking extensively. We chose the median optical flow because it shares information between its separate instances when tracking different image areas in the same video sequence, which makes it relatively fast (see Section 2.2.4). However, usage of different tracking algorithms is possible. We write $(x', y') = F_{t_1 t_2}(x, y)$ for the result of a tracking algorithm from frame t_1 to frame t_2 applied to the image region centered at (x, y) with the size of a body part. Furthermore, $\hat{F}_{t_1 t_2}(p^{t_1}) = \{F_{t_1 t_2}(p_i^{t_1})\}_{i=1}^{K}$ is the pose obtained by tracking pose p^{t_1} to frame t_2.

Poses search set. P determines the set of poses considered in (4.1). Let $\mathrm{FMP}(I_t)$ denote a set of n best-scoring poses, returned by flexible mixture of parts for frame t. We first populate P with $\mathrm{FMP}(I_t)$ and then expand it

with tracking FMP(I_t) with median optical flow θ frames back and forward. We find expansion necessary as it usually fills in the correct poses, missing from FMP(I_t):

$$P = \bigotimes_{t=1}^{T} \left(\text{FMP}(I_t) \cup \bigcup_{\tau=\tau_1}^{\tau_2} \{\hat{F}_{\tau t}(p^\tau) | p^\tau \in \text{FMP}(I_\tau)\} \right), \tag{4.3}$$

where $\tau_1 = \max(1, t-\theta)$, $\tau_2 = \min(T, t+\theta)$ and $\bigotimes$ denotes Cartesian product.

Local scores. $S_{loc}(I_t, p^t)$ define the local score of pose p^t in image I_t. Since we are determined to use HOG features only, the best score of the pose would be the actual score returned by FMP, as it contains both the appearance and deformation parts. However, there is no score assigned to many poses in P, as they were obtained by tracking, as opposed to be directly returned by FMP. We reconstruct the scoring function of FMP by computing the best combination of filters given current position of body parts:

$$S_{\text{FMP}}(I_t, p^t) = \max_{f^t} S(I_t, p^t, f^t), \tag{4.4}$$

where $S(I, p, f)$ is defined as in (2.2.1). This can be done using dynamic programming with the following message passing:

$$score_i^t(f_i^t, p^t) = b_i^{f_i^t} + \omega_i^{f_i^t} \cdot \phi(I_t, p_i^t) + \sum_{j \in kids(i)} m_j^t(f_i^t, p^t), \tag{4.5}$$

$$m_j^t(f_i^t, p^t) = \max_{f_j^t} \left(b_{ji}^{f_j^t f_i^t} + score_j^t(f_j^t, p_j^t) + \omega_{ji}^{f_j^t f_i^t} \cdot \psi(p_j^t - p_i^t) \right). \tag{4.6}$$

FMP is far from being perfect, and it frequently happens that low-scoring body configurations estimate the pose better than high-scoring ones. With the above scoring function correct detections that were obtained by tracking and not selected by FMP will have low scores and will often be rejected by the dynamic programming algorithm selecting the best-overall combination of poses p. However, the tracking origins of such poses as direct outputs of the FMP will score higher. This motivates us to alter the local scores of tracked poses to capture both the score of the tracking origin and the actual

score by blending them together:

$$S_{loc}(I_t, p^t) = \begin{cases} S_{\text{FMP}}(I_t, p^t) & \text{if } p^t \in \text{FMP}(I_t), \\ |\frac{\tau}{\theta}| S_{\text{FMP}}(I_t, p^t) & \\ +(1 - |\frac{\tau}{\theta}|) S_{\text{FMP}}(I_{t+\tau}, \hat{F}^{-1}_{t+\tau,t}(p^t)) & \text{if } p^t \in \hat{F}_{t+\tau,t}(\text{FMP}(I_{t+\tau})). \end{cases} \tag{4.7}$$

Pairwise scores. $S_{pair}(I_{t-1}, I_t, p^{t-1}, p^t)$ represents score between poses p^{t-1}, p^t in two adjacent frames. We use squared euclidean distance between the pose obtained by tracking of p^{t-1} and pose p^t:

$$S_{pair}(I_{t-1}, I_t, p^{t-1}, p^t) = Cd(\hat{F}_{t-1,t}(p^{t-1}), p^t), \tag{4.8}$$

$$d(p^{t_1}, p^{t_2}) = \sum_{i=1}^{K} (x_i^{t_1} - x_i^{t_2})^2 + (y_i^{t_1} - y_i^{t_2})^2, \tag{4.9}$$

where C is a normalizing constant utilized in order to make the local and pairwise scores comparable.

4.2 Inference

Inference corresponds to maximizing (4.2) with respect to the combination of poses p. This can be done efficiently using dynamic programming with the following message passing:

$$score_t(p^t) = S_{loc}(I_t, p^t) + \max_{p^{t-1} \in P_{t-1}} S_{pair}(I_{t-1}, I_t, p^{t-1}, p^t), \tag{4.10}$$

where $P_t = \{p_t | (p_1, \ldots, p_t, \ldots, p_T) \in T\}$ is a set of poses in frame t. After passing messages throughout the whole chain of poses $score_T(p^T)$ would contain the total scores of pose configurations. The final set of poses can be obtained by taking the maximum-scoring pose from $score_T(p^T)$ and applying backtracking.

The inference procedure can be summarized as follows:

Input: Set of images $I = \{I_t\}_{t=1}^{T}$, constants n, θ
Output: Set of poses $p = (p^1, \ldots, p^T)$
for *each frame* I_t **do**
  $\hat{P}_t \leftarrow$ the set of n best poses returned by FMP;
  if $t > 1$ **then**
    $f_{t-1,t}(x, y) \leftarrow$ optical flow for all (x, y);
  end
end
for *each frame* I_t **do**
  $P_t \leftarrow \hat{P}_t$;
  $\tau_1 \leftarrow \max(1, t - \theta)$;
  $\tau_2 \leftarrow \min(T, t + \theta)$;
  for $\tau = \tau_1, \ldots, \tau_2, \tau \neq t$ **do**
    for *each* $p^\tau \in \hat{P}_t$ **do**
      $F_{\tau t}(p_i^\tau) \leftarrow$ median of optical flow around p_i^t;
      $p^t \leftarrow \{F_{\tau t}(p_i^\tau)\}_{i=1}^{K}$;
      $S_1 \leftarrow$ reconstructed score of FMP;
      $S_2 \leftarrow$ score of p^τ returned by FMP;
      $\alpha \leftarrow |t - \tau|/\theta$;
      $S_{loc}(I_t, p^t) \leftarrow \alpha S_1 + (1 - \alpha) S_2$;
      $P_t \leftarrow P_t \cup p^t$;
    end
  end
end
for *each frame* $I_t, t > 1$ **do**
  for *each* $p^{t-1} \in P_{t-1}$ **do**
    for *each* $p^t \in P_t$ **do**
      $S_{pair}(I_{t-1}, I_t, p^{t-1}, p^t) \leftarrow d(\hat{F}_{t-1,t}(p^{t-1}), p^t)$
    end
  end
end
$(score(p), ind(p)) \leftarrow$ dynamic programming on S_{loc}, S_{pair};
$(p^1, \ldots, p^T) \leftarrow$ backtracking with $ind(\arg\max(score))$;

Algorithm 1: Pose estimation in video procedure.

4.3 Experiments

Parameter Adjustment. The major parameters in the algorithm to be set are the number of top FMP detections in each frame n and the number of frames each original detection is tracked back and forward to θ. The major factor that we take into account when adjusting parameters for the model is the time of inference. We would like the computation time of our method to be of the same order of magnitude as FMP. The latter often runs in 5-20 seconds for a single image on a conventional machine. We limit ourselves to 60 seconds per image frame, which puts certain constraints on n and θ.

The computation time during detection is mostly consumed by two stages: computing n FMP detections and tracking detection back and forward θ frames. The FMP computation time does not depend on n, therefore we have to minimize the tracking time, which in our experiments takes on average 80% of the inference time. In every frame our algorithm performs articulated tracking $2n\theta$ times, which involves tracking each of the 26 body parts in the model. We use median optical flow, which on average takes 0.06 seconds per body part, taking into account pre-computation of optical flow for every consecutive pair of images. Therefore we impose a constraint $n\theta < 20$ in order to satisfy 60 seconds per frame computation time requirement.

Although the median optical flow is fast, it is not the best tracking algorithm. In our experiments we observe that tracking for more than 2 frames often drifts away and picks the background regions especially when people in the videos move fast. Therefore we set $\theta = 2$ to maximize the number of FMP detections, and set $n = 10$. Our early experiments demonstrated the benefit of this approach in comparison with $\theta = 5$, $n = 4$. We also set the transition cost weighting constant $C = 1$, as it does not seem to significantly affect the results.

Evaluation. We evaluate our algorithm using PCP2D, a definition of percentage of correct parts proposed in Section 3.2.1. As discussed earlier, evaluation of an algorithm that does not distinguish between left and right sides of a body on a dataset that does differentiate them using standard PCP measure does not necessarily reflect the performance of the algorithm. We use PCP2D based on a more common (loose) version of PCP (see Section 3.2.1). The threshold for the PCP measure often varies depending on the dataset. Since our detections are 26-part body skeletons, the parts themselves are smaller than in a 14-joint skeleton, therefore we set the threshold to a relatively large value 0.6.

Many videos in the UCF Sports Pose dataset contain multiple people

Table 4.1: Results of our shortest path approach (SPA). Results are compared to FMP [55] for different action classes and overall.

Action Class	Golf Swing	Kick	Lift	Ride Horse	Run	Skate	Walk	All
FMP [55]	58%	57%	72%	52%	52%	58%	68%	60%
SPA	**64%**	**60%**	**82%**	**60%**	**60%**	**64%**	**79%**	**68%**

either in the background or performing similar actions together. FMP is a detection approach, which may often detect people other than the target person. In order to make the comparison to FMP fair we crop the initial video sequences such that they contain single person only. Due to the time constraints we evaluate our algorithm on a subset of UCF Sports Pose dataset, containing 4 to 10 videos per action class, totaling from 230 to 405 video frames per action class. It has 39 video sequences with 2305 video frames in total. The comparison of the algorithm introduced in this chapter and FMP for each action class and overall is presented in Table 4.1. For the examples refer to Figures 4.1-4.3. The author's website[1] provides several video examples comparing FMP to our approach.

4.4 Discussion and Future Work

Comparison to other methods. The approach described in this chapter is close in some of its ideas to both [51] and [58], although it was developed independently, as its development started before these papers were published. Here we would like to briefly contrast these methods with our approach.

Both our approach and [51] take the top n outputs of the FMP. Wang *et al.* [51] find the best-overall combination of poses using dynamic programming. They score every pose according to a pre-learned color model, and score co-occurrences of poses in adjacent frames using color similarity. We find the best combination of poses in the same way, however we chose not to use any features other than HOG. We want to make the comparison to FMP fair, and we are interested in seeing how the addition of only flow information improves the detections. Instead, we use the score returned by the FMP itself. In order to obtain the pairwise scores we propagate the

[1] `http://www.cs.ubc.ca/nest/lci/thesis/olgeorge/index.html`

body poses to the adjacent frames and compute their displacements, while Wang *et al.*do not use optical flow.

We further decide to incorporate knowledge from adjacent frames in every frame. We propagate the poses from adjacent frames to the current frame and add them to the pool of poses for dynamic programming. In this we are similar to Zuffi *et al.* [58] who propagate poses from the neighbouring frames in order to aggregate more information for further processing.

Future work. Our algorithm "fails" most often in the presence of fast motion or motion blur. The tracking of poses loses the body parts, resulting in incorrect propagation of information, which often causes wrong pose estimates. In order to improve our algorithm one may replace the median optical flow with a more accurate tracker. As the tracker is required to be relatively fast, we foresee two potential candidates. The first one called Median Flow [27] combines forward-backward error filtering and normalized cross-correlation. It is based on optical flow and may satisfy the speed requirement. The second alternative to median optical flow is the utilization of trajectories, such as dense trajectories [52]. One may compute the trajectories for the current frame and then use the median value in a box for tracking. This procedure should not be time consuming as well. In order to improve the pairwise scoring of poses in adjacent frames one may utilize learning of the co-occurrence patterns of the appearance features of the model. An alternative direction of the future work is to make the algorithm online, as many applications of pose estimation such as human-robot interaction require the processing of the information on-the-fly. This could be achieved by utilizing a hidden Markov model instead of the Viterbi algorithm when looking for the best-overall combination of poses.

Our approach may be considered as a sampling method that first choses a subset of all possible points representing the space of its model and then does the full search on the results. The performance of a sampling method depends on the sampling technique, which in our case is heavily based on FMP. Therefore, it has very strict limitations to the set of poses it can produce, which is determined by the output of FMP. Thus, if on a certain sequence FMP fails, our method would fail as well. Furthermore, our method would make the largest improvement on the sequences where FMP works sufficiently well to detect the right pose among its top candidates, but not well enough to pick the best one. In other words, our approach allows FMP to fix itself based on what it already knows, filtering out incorrect detections. Also, our method knows nothing about the dynamics of human motion, performing only spatial reasoning about the discrepancies in consecutive frames of a video sequence. In the next chapter we aim to address the above

issues by introducing a method that does full search over several consecutive frames while taking into account the change of the appearance in time.

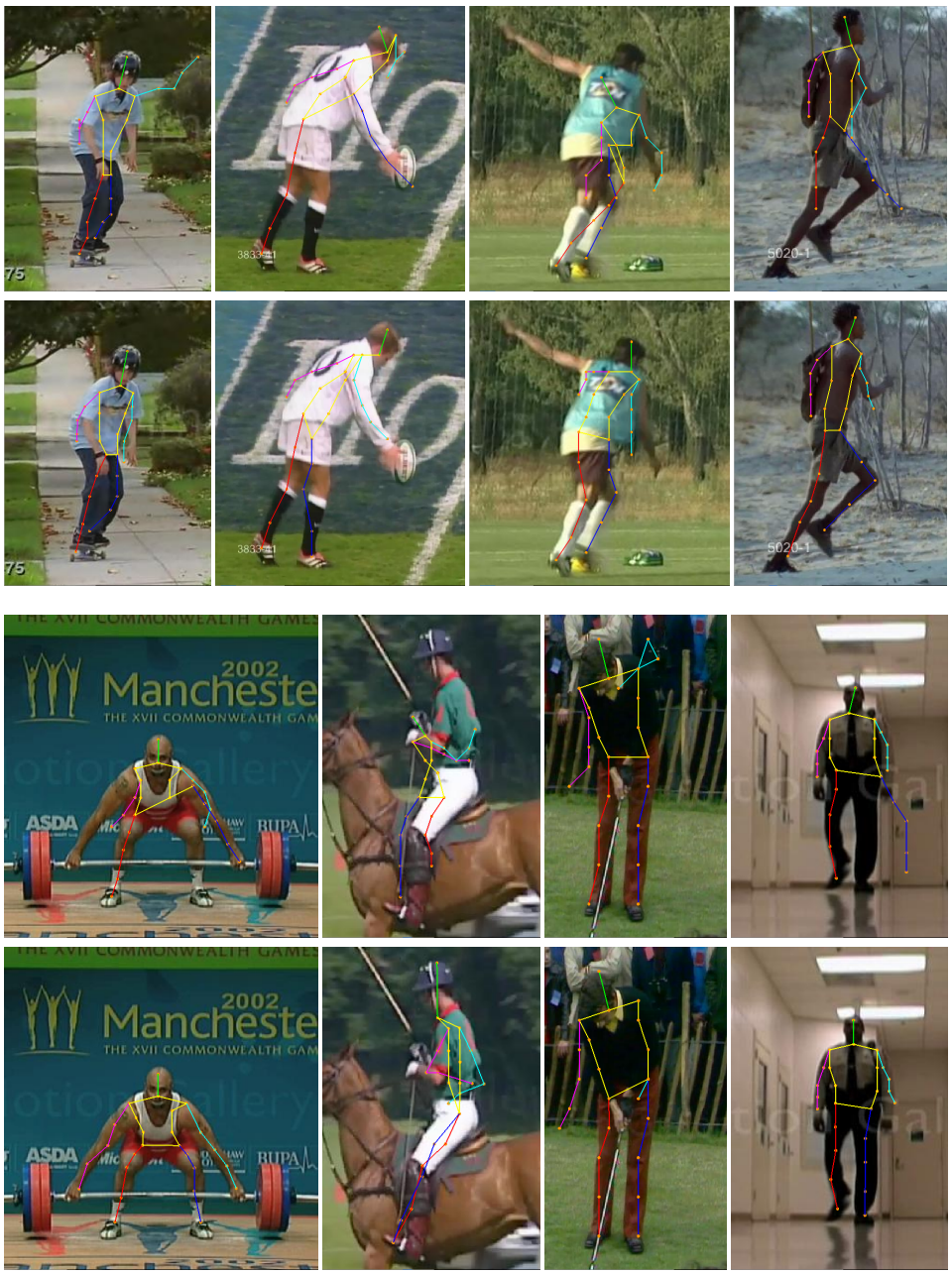


Figure 4.1: Examples of pose estimates of the shortest path approach. The results are compared to the results of FMP [55]. The first and third rows contain the results of FMP, the second and fourth rows contain the results of our method on the same images.

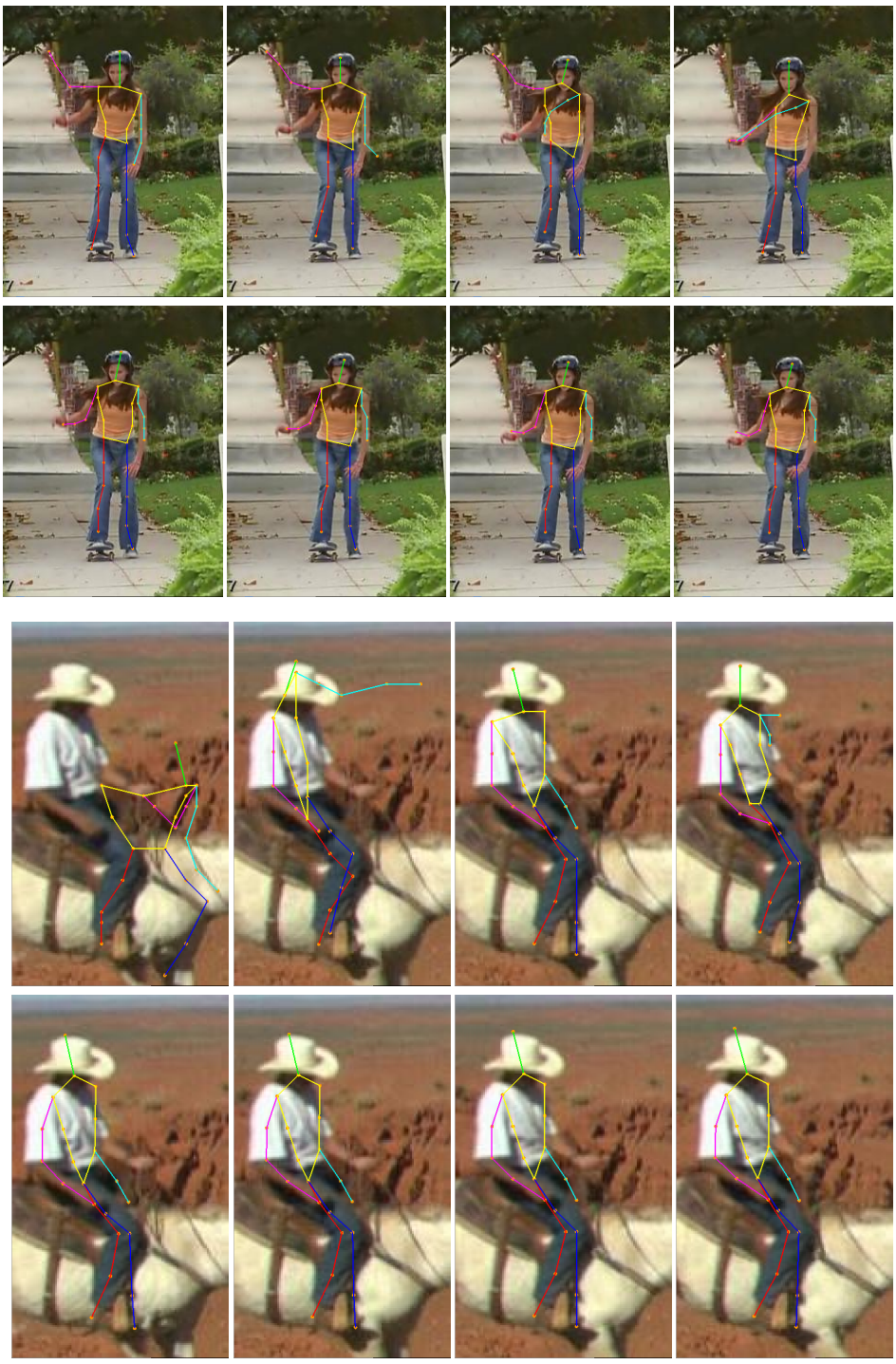


Figure 4.2: Examples of pose estimates of the shortest path approach. The results are compared to the results of FMP [55]. The first and third rows contain the results of FMP, the second and fourth rows contain the results of our method on the same images.

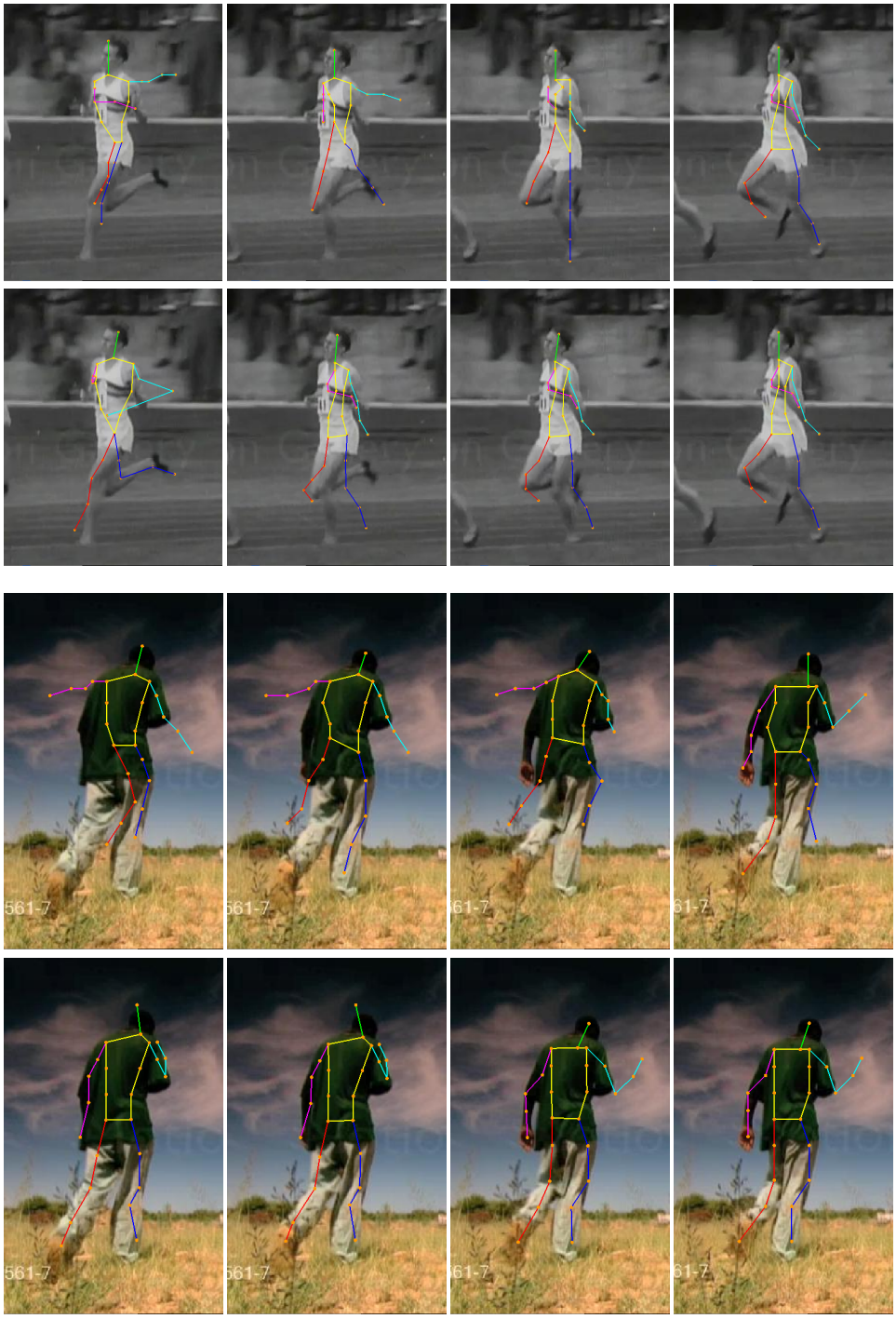


Figure 4.3: Examples of pose estimates of the shortest path approach. The results are compared to the results of FMP [55]. The first and third rows contain the results of FMP, the second and fourth rows contain the results of our method on the same images.

Chapter 5

Pose Estimation in Video: a Detection Approach

The approach described in Chapter 4 gives better results than the original Flexible Mixture of Parts (FMP). Acting as a smoothing filter, it helps to get rid of sporadic incorrect detections, giving an overall better estimation of pose throughout the video sequence. However, it is fundamentally a filtering approach, and as was discussed in Section 4.4 it can give better results only when the original approach succeeds more often than fails.

In order to address the above limitation in this chapter we introduce a novel articulated human detection algorithm in video sequences. In contrast to the previous approach that searches only among the best FMP results, it is a detection algorithm that enables full search in several consecutive frames at once, which internally takes into account both appearance and motion information. Like FMP, it utilizes a tree model, allowing fast and tractable inference using dynamic programming and a modified distance transform. In addition it is an online method, in the sense that at every point in time it does not require any future information in order to detect a pose in the current frame. The latter makes it possible to run it in real-time, given enough computational power.

This chapter is organized as follows. In Section 5.1 we define the model, in Section 5.2 we describe inference algorithm together with our modification of the distance transform of sampled functions. We present results in Section 5.3.

5.1 Model

The main idea behind the current method is the way to incorporate motion and appearance in a single model, such that it captures how the appearance changes with time. This can be achieved by learning the co-occurrence patterns of filter types, corresponding to the same body part in consecutive frames. Thus, the model fully connecting several tree models covering the

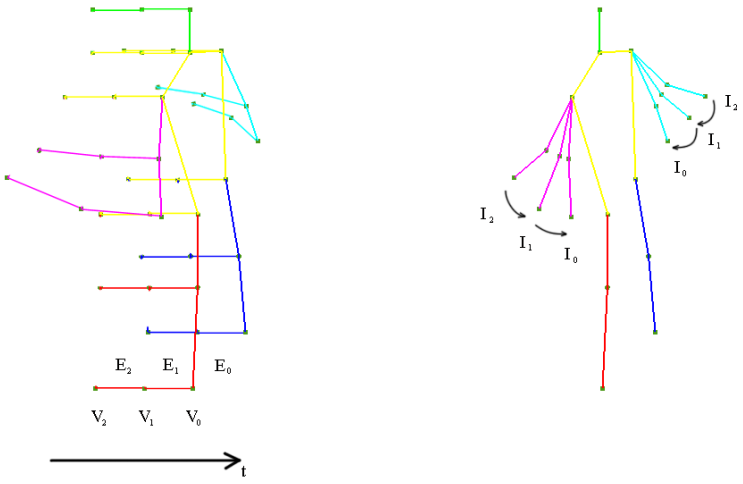
human body in adjacent frames may be utilized. However, inference in such model becomes intractable. In order to restore the tree property of the model graph, we drop the limb connections in all frames except the first one, leaving a "trail" of positions in the past several frames for every part (see Figure 5.1). Although the dropped connections would distort the positions of body parts in previous frames, this may be compensated by the temporal connections, aligned with the optical flow. Furthermore, the inference problems that arise when utilizing this model can be solved by a modified distance transform.

Let us write $I = \{I_t\}_{t=0}^{\Theta}$ for a sequence of $\Theta + 1$ video frames, where I_0 is the frame where we want to detect a pose and $I_\Theta \dots I_1$ are Θ preceding frames. We use descending enumeration for convenience. Our model utilizes a tree graph $(V, E) = (\bigcup_{t=0}^{\Theta} V_t, \bigcup_{t=0}^{\Theta} E_t)$ spanning $\Theta+1$ frames, such that in frame I_0 graph (V_0, E_0) represents a K-node tree model of the human body, while nodes $V_\Theta \dots V_1$ correspond to locations of body parts in Θ preceding frames and edges $E_\Theta \dots E_1$ connect body parts to their instances in the previous frame (See Figure 5.1). Formally, we use double indexing for nodes in the graph, such that $V = \{(i,t)\}_{i=1,t=0}^{K,\Theta}$ where t denotes the frame number, and i represents the body part index. Then $E_0 = \{((i,0),(j,0))\}$ and $E_t = \{((i,t-1),(i,t))\}_{i=1}^{K}$. For convenience we use the following notation: $V'_t = \{i\}_{i=1}^{K}$, $E'_0 = \{(i,j)\}$, $E'_t = \{i\}_{i=1}^{K}$.

Furthermore, let $p_i^t = (x_i^t, y_i^t)$ be the pixel location of body part i in frame I_t. Then $p^t = \{p_i^t\}_{i=1}^{K}$ defines all body part locations in frame I_t, and $p = (p^0, \dots, p^\Theta)$ is the total spatial configuration of body parts in $\Theta + 1$ frames. Likewise, let $f_i^t = \{1, \dots, R\}$ determine the filter type for body part i in frame t, then $f^t = \{f_i^t\}_{i=1}^{K}$ and $f = (f^0, \dots, f^\Theta)$ represent the filter configuration in frame t and the total filter configuration correspondingly. Also, similarly to Section 4.1 let $(x', y') = F_t(x, y), t = \{1, \dots, \Theta\}$ denote the median optical flow frame $t-1$ to frame t in the image region centered at (x, y) with the size of a body part (see Section 2.2.4).

The score of a specific configuration of body part locations p and filter types f in $\Theta + 1$ video frames I has the following form:

$$S(I, p, f) = \sum_{t=0}^{\Theta} S_t(I, p, f), \tag{5.1}$$



(a) (b)

Figure 5.1: Spatio-temporal tree structure of the model. Graph (V_0, E_0) represents body structure in the frame where detection is being performed. Each set of nodes V_t correspond to locations of body parts t frames back in time, each set of edges E_t connects nodes in V_{t-1} to their corresponding nodes in V_t. (a) An example of the model structure for $\Theta = 2$. (b) Corresponding poses for frames I_2, I_1, I_0.

$$\begin{aligned} S_0(I,p,f) = &\sum_{i\in V_0'} b_i^{f_i^0} + \sum_{(i,j)\in E_0'} b_{ij}^{f_i^0 f_j^0} + \dots \\ &\sum_{i\in V_0'} \omega_i^{f_i^0}\cdot\phi(I_0,p_i^0) + \sum_{(i,j)\in E_0'} \omega_{ij}^{f_i^0 f_j^0}\cdot\psi(p_i^0 - p_j^0), \end{aligned} \tag{5.2}$$

$$\begin{aligned} S_t(I,p,f) = &\sum_{i\in V_t'} b_i^{f_i^t} + \sum_{i\in E_t'} b_i^{f_i^t f_i^{t-1}} + \dots \\ \sum_{i\in V_t'} \omega_i^{f_i^t}\cdot\phi(I_t,p_i^t) + &\sum_{i\in E_t'} \omega_i^{f_i^t f_i^{t-1}}\cdot\psi(p_i^t - F_t(p_i^{t-1})), \quad t=\{1,\dots,\Theta\}. \end{aligned} \tag{5.3}$$

In the above equation $\phi(I_t, p_i^t)$ is a feature vector extracted from image I_t at location p_i^t. This could be a HOG descriptor [9] or any other feature. We also write $\psi(d_x, d_y) = [d_x^2 \quad d_y^2 \quad d_x \quad d_y]^\Theta$.

Given this notation, our model has the following form:

$$M = (B, \Omega), \tag{5.4}$$
$$B = \left(\{b_i^m\}, \{b_{ij}^{mn}\}, \{b_i^{mn}\}\right), \tag{5.5}$$
$$\Omega = \left(\{\omega_i^m\}, \{\omega_{ij}^{mn}\}, \{\omega_i^{mn}\}\right). \tag{5.6}$$

Here b_i^m favours assignment of filter type m to body part i, b_{ij}^{mn} favours co-occurrence of filter types m and n in body parts i and j, b_i^{mn} favours switching from filter m to filter n in body part i in two consecutive frames. Furthermore, ω_i^m is the filter of type m of body part i. The quadratic deformation spring model between filters m and n of body parts i and j is determined by ω_{ij}^{mn}. Also, ω_i^{mn} defines the deformation model for the switching from filter n to filter m of body part i in two consecutive frames.

Note that $S_0(I, p, f)$ is exactly the cost function of the Flexible Mixture of Parts [55] (see Section 2.2.1). Thus, our model turns into FMP in the case when $\Theta = 0$.

5.2 Inference

Inference corresponds to maximizing the model's score function (5.1) over parameters (p, f) given a sequence of video frames $I = \{I_t\}_{t=0}^{\Theta}$. Since our relational graph (V, E) is a tree, dynamic programming enables full search over all possible locations p and filter types f, similarly to FMP.

5.2.1 Message Passing

In order to perform dynamic programming, we set up a message passing mechanism from child to parent nodes (see Section 2.2.2). Let $kids(i,t)$ be the set of children of node (i,t), let $kt(i,t)$ denote the temporal child of node (i,t) and $ks(i,t)$ denote its spatial children:

$$ks(i,t) = \{(j,\tau) \in kids(i,t) | \tau = t\},$$
$$kt(i,t) = \{(j,\tau) \in kids(i,t) | \tau = t+1, j = i\}.$$

The message that child (i,t) passes to its parent has the following form:

$$\begin{aligned} score_i^t(f_i^t, p_i^t) = b_i^{f_i^t} + \omega_i^{f_i^t} \cdot \phi(I_t, p_i^t) + \dots \\ + \sum_{(j,\tau)\in ks(i)} ms_j^t(f_i^t, p_i^t) + \sum_{(j,\tau)\in kt(i)} mt_i^{t+1}(f_i^t, p_i^t), \end{aligned} \tag{5.7}$$

where ms_j^t and mt_i^t are defined as follows:

$$ms_j^t(f_i^t, p_i^t) = \max_{f_j^t} \left[b_{ji}^{f_j^t f_i^t} + \max_{p_j^t} \left(score_j^t(f_j^t, p_j^t) + \omega_{ji}^{f_j^t f_i^t} \cdot \psi(p_j^t - p_i^t) \right) \right], \tag{5.8}$$

$$mt_i^t(f_i^{t-1}, p_i^{t-1}) = \max_{f_i^t} \left[b_i^{f_i^t f_i^{t-1}} + \max_{p_i^t} \left(score_i^t(f_i^t, p_i^t) + \omega_i^{f_i^t f_i^{t-1}} \cdot \psi(p_i^t - F_t(p_i^{t-1})) \right) \right] \tag{5.9}$$

Note that our inference procedure is obtained from the one in FMP by adding the second sum in equation (5.7), which has 0 and 1 terms for leaves and internal nodes correspondingly.

The message passing starts from leaves, and proceeds until all the nodes except the root have passed messages to their parents. Then, $score_1^0(f_1^0, p_1^0)$ contains the final scores for detection, and similar to FMP we obtain multiple detections by thresholding the score and applying non-maximum suppression (see Section 2.2.1) on the result to remove the detections covering the same human. We use backtracking to restore the detected poses (see Section 2.2.2).

5.2.2 An Approximate Distance Transform

The computationally expensive portion of message passing is computing (5.8) and (5.9). It requires looping over $L \times R$ possible locations and types of the parent and $L \times R$ potential locations and types of the child, making the complexity of the whole procedure $O(L^2R^2)$. The relaxation (2.2) reduces the complexity to $O(L^2R)$. However, given that the total number of possible locations L is often very large, the quadratic complexity makes inference procedure too slow, almost impractical. In our experiments it took more than an hour to find a pose in a small image on a conventional PC.

Therefore, utilization of methods reducing computation time is essential. Yang and Ramanan [55] use the distance transform developed by Felzenszwalb and Huttenlocher [15], which reduces computation of (5.8) to $O(LR^2)$ in the case when $\psi(d_x, d_y)$ is a quadratic function (see Section 2.2.3). However, direct usage of the aforementioned distance transform for computation of (5.9) is not possible. In this section we will describe how one can modify it for our case.

One dimension. Consider the following problem. Let $G = \{g_1, \ldots, g_n\}$ and $H = \{h_1, \ldots, h_m\}$ be one-dimensional grids, $f : G \to \mathbb{R}$ and $d : H \to \mathbb{R}$ be arbitrary functions. For every p in grid H find $D_f(p)$, defined as:

$$D_f(p) = \min_{q \in G} P_2(p + d(p), q), \tag{5.10}$$

$$P_2(p, q) = a(p - q)^2 + b(p - q) + f(q) \tag{5.11}$$

This problem can be reduced to Felzenszwalb's distance transform of sampled functions [15] in the following way. First, we perform the computation of the lower envelope of parabolas $P_2(p, q)$ for all q in G. Then we fill in the values of $D_f(p)$, but we replace p with $p + d(p)$ when computing values of the lower envelope.

Two dimensions. Let $f : G \to \mathbb{R}$ be an arbitrary function defined on two-dimensional grid $G = \{g_1^1 \ldots g_n^1\} \times \{g_1^1 \ldots g_1^k\}$, and let $d : H \to \mathbb{R}$ be an arbitrary function defined on grid $H = \{h_1^1 \ldots h_m^1\} \times \{h_1^1 \ldots h_1^l\}$. The goal is to find $D_f(x, y)$ for every (x, y) in H:

$$D_f(x, y) = \min_{(x', y') \in G} P_2((x, y) + d(x, y), (x', y')), \tag{5.12}$$

$$\begin{aligned} P_2((x, y), (x', y')) = a_x(x - x')^2 + b_x(x - x') \\ + a_y(y - y')^2 + b_y(y - y') + f(x', y'). \end{aligned} \tag{5.13}$$

$$D_f(x, y) = \min_{(x', y') \in G} a_x(x + d_1(x, y) - x')^2 + a_y(y + d_2(x, y) - y')^2 + f(x', y') \tag{5.14}$$

In the case when $d(x, y) \equiv 0$ the problem reduces to (2.17)-(2.18), which can be formulated as (2.19). The latter can be solved by performing Felzenszwalb's one-dimensional distance transform along each column of the grid G and then computing the distance transform along each row of the result. Similar reduction to the one-dimensional case (5.10) is possible when the function $d(x, y) = (d_1(x, y), d_2(x, y))$ satisfies the constraints

$$d_1(x, y_1) = d_1(x, y_2) = d_1(x), \qquad \forall x \in \{h_1^1 \ldots h_n^1\}, \tag{5.15}$$

$$d_2(x_1, y) = d_2(x_2, y) = d_2(y), \qquad \forall y \in \{h_1^1 \ldots h_1^k\}, \tag{5.16}$$

because in this case the first two terms of (5.13) do not depend on y:

$$D_f(x,y) = \min_{x',y'}(a_x(x+d_1(x,y)-x')^2 + b_x(x+d_1(x,y)-x')+$$
$$a_y(y+d_2(x,y)-y')^2 + b_y(y+d_2(x,y)-y') + f(x',y')) = \tag{5.17}$$
$$\min_{x'}[a_x(x+d_1(x)-x')^2 + b_x(x+d_1(x)-x')+$$
$$\min_{y'}(a_y(y+d_2(y)-y')^2 + b_y(y+d_2(y)-y') + f(x',y'))] = \tag{5.18}$$
$$\min_{x'}[a_x(x+d_1(x)-x')^2 + b_x(x+d_1(x)-x') + D_f|_{x'}(y)]. \tag{5.19}$$

Intuitively, the aforementioned reduction is possible because the set of points $(x,y)+d(x,y)$ forms a grid:

$$R = \big\{(x,y)+d(x,y)|(x,y)\in H\big\} \equiv \{r_1^1 \dots r_m^1\} \times \{r_1^1 \dots r_1^l\}. \tag{5.20}$$

However, in the general case when equalities 5.15-5.16 do not hold the procedure outlined above does not provide the solution to the problem 5.10-5.11. One of the possible ways to deal with it is to form a grid from the set R as defined in (5.20) by taking the Cartesian product of its projections on the X and Y axes. This however may expand the set from L to L^2 points and although in this case the above procedure could be utilized, the computation will not be performed in linear time, thus the benefit of the distance transform will be lost.

An alternative solution to this problem would be a distance transform working directly in two dimensions by utilizing a two-dimensional lower envelope of elliptic paraboloids. However, the computation of the lower envelope in two dimensions is a much more complicated procedure, as one has to find intersections of every elliptic paraboloid with all its neighbours, and potentially neighbours of the neighbours, forming a complex partition of the two-dimensional space, consisting of polygons of potentially arbitrary shape.

We take a different approach. We quantize the set R as defined in (5.20) into grid H, by obtaining $\tilde{d}(x,y)$ such that $(x,y)+\tilde{d}(x,y)$ is the closest point to $(x,y)+\tilde{d}(x,y)$ in grid H. Then we use the distance transform in the conventional way, obtaining $D_f(x,y)$ as defined in (2.17). Finally we compute the approximation to the distance transform $\tilde{D}_f(x,y)$ as defined in (5.14):

$$\tilde{D}_f(x,y) = D_f(x+\tilde{d}_1(x,y), y+\tilde{d}_2(x,y)). \tag{5.21}$$

The above is equivalent to quantizing the optical flow information to the nearest HOG cell. Since body part filters are often represented by 4×4 to 6×6 HOG grids, this loss of information is not dramatic, which may be effectively mitigated for a more accurate tracking algorithm. Furthermore, when the quadratic coefficients a_x, a_y are sufficiently small as in our experiments, the paraboloids are wide, and the difference in the computation of the score is minimal.

In order to apply the distance transform to message passing (5.9), one has to define $[a_x, a_y, b_x, b_y] \equiv \omega_i^{f_i^t f_i^{t-1}}$, $f(x,y) = score_i^t(f_j^t, (x,y))$, $d(x,y) = F_t(x,y)$. Then utilization of the approximate distance transform is possible, reducing the corresponding portion of message passing to linear time.

5.2.3 The Inference Procedure

As mentioned in the beginning of this chapter, the approach described above is a detection algorithm, requiring information only about previous frames. Given a video sequence $V = \{V_1, \ldots, V_T\}$ the inference corresponds to finding a pose in frame t for all $t \in \{\tau\Delta + 1, \ldots, T\}$, taking into account only video frames $V_1, \ldots, V_t$. In order to do this, for every frame t we select Θ previous frames with temporal step Δ:

$$I = \{I_\tau\}_{\tau=0}^{T}, \tag{5.22}$$

$$I_\tau = V_{t-\tau\Delta}, \quad \tau = \{0, \ldots, \Theta\}. \tag{5.23}$$

Then, for every t the inference procedure described in this section can be performed independently, obtaining a full temporal tree, representing the human body in Θ frames. The temporal part can be then disregarded, leaving only the spatial part in the current frame, which represents the final result of the detection.

Input: Sequence of video frames $V = \{V_1, \ldots, V_T\}$, constants Θ, Δ
Output: Set of poses $p = (p^1, \ldots, p^T)$
for *each* $t \in \{\tau\Delta + 1, \ldots, T\}$ **do**
  for *each* $\tau \in \{0, \ldots, \Theta\}$ **do**
    $I_\tau \leftarrow V_{t-\tau\Delta}$;
  end
  $p \leftarrow \arg\max_p \max_f S(\{I_\tau\}_{\tau=0}^{\Theta}, p, f)$;
  $p^t \leftarrow (p)_1$;
end

5.3 Experiments

Parameter Adjustment. Our model consists of a spatial and a temporal part. The spatial part defines how well the model fits the current frame, while the temporal part identifies how consistent the appearance and location of every body part in time is. We think that $b_i^{f_i^t}$ and $\omega_i^{f_i^t}$ from equalities 5.1-5.1 that determine appearance bias and filter can be learned independently of the temporal dimension, because the detection is performed in evey frame independently taking into account several previous frames, and these parameters may not depend on t. Therefore we use the corresponding parameters from the FMP model trained on single images: $b_i^{f_i^t} = b_i^{f_i^0}$ and $\omega_i^{f_i^t} = \omega_i^{f_i^0}$. We follow Yang and Ramanan [55] and resort to relaxation $\omega_{ij}^{f_i^0 f_j^0} = \omega_{ij}^{f_i^0}$, which reduces the computation cost during inference.

We explore two simple methods for learning $b_i^{f_i^t f_i^{t-1}}$. The first method finds the score of every ground truth pose in the training video sequence by fixing the position and maximizing over all possible filter types for all body parts using equations 4.5-4.6. Then for each body part it counts co-occurrences of different filter types in two consecutive frames. The second method acts similarly, but it takes into account all filter responses at every time instance, as opposed to only counting the filter types that were selected in the score reconstruction process. If in the current frame filter type i has score s_i, and in the next frame filter type j has score s_j, then the co-occurrence bias of filter types i and j is increased by $s_1 s_2$. When both i and j either score high or negatively low this favours co-occurrence of these filter types. It penalizes the co-occurrence when one of them is high and the other one is low. We also explored other counting functions such as $s_1 + s_2$, $\max(s_1, 0) \max(s_2, 0)$ but found no difference in several early experiments.

Although the above learning schemes demonstrated efficiency in early experiments, we found that they do not always improve the pose estimation performance even on the videos they were trained on, and do not translate well between video sequences. Therefore in our experiments we set $b_i^{f_i^t f_i^{t-1}} = 0$ and explore the framework in the absence of the appearance switch bias. We set $\omega_i^{f_i^t f_i^{t-1}} = \omega_i^{f_i^0}$ as it does not significantly alter the results. Such an approach does not require any training video data, as we use the model trained on single images. We set the temporal step $\Delta = 2$ to increase the discrepancy between two consecutive frames and set the number of simultaneously considered frames in the past as $\Theta = 1$. Increasing Θ above 2 results in the growth of the number of tracking failures of our

Table 5.1: Results of our detection approach (DA). Results are compared to FMP [55] for different action classes and overall.

Action Class	Golf Swing	Kick	Lift	Ride Horse	Run	Skate	Walk	All
FMP [55]	**58%**	**57%**	72%	52%	52%	58%	**68%**	60%
DA	57%	53%	**73%**	**54%**	**58%**	**58%**	67%	**62%**

median optical flow algorithm. This decreases the algorithm performance, as the model is constrained to search for the pose using incorrect spatio-temporal tree configuration.

Evaluation. Similarly to Section 4.3 we evaluate the algorithm using PCP2D on the same subset of videos from the UCF Sports Pose dataset, consisting of 29 video sequences totaling 2305 video frames. The comparison of the algorithm to FMP for each action class and overall is demonstrated in Table 5.1. See Figures 5.2-5.3 for some pose estimation examples.

5.4 Discussion and Future Work

As one may see from Figures 5.2-5.3 our algorithm gives results very close to the results of FMP. The optical flow does add new information which leads to overall marginal improvement. However, on certain video sequences our method works worse than FMP because inaccurate tracking imposes incorrect priors on a pose in the previous frame.

We see several steps that may address the issues above. The first one utilizes a joint spatio-temporal training scheme similarly to Yang and Ramanan [55]. It may help define the biases $b_i^{f_i^t f_i^{t-1}}$ more optimally, which may play an important role in the detection process by penalizing unlikely filter switches. For instance, the horizontal forearm in the current frame is unlikely to be vertical in the next frame as the change is too abrupt. Such relations may be captured by a jointly learned spatio-temporal model.

However, providing a better learning scheme may not be enough to make a substantial improvement in accuracy. Since we relax the spatial relationships between joints in previous frames we need a good prior on the location of these joints. The second step that we suggest for future work is to seek a better tracking mechanism. As discussed in Section 4.4 the potential algorithms are Median Flow [27] and dense trajectories [52].

An alternative way to improve the performance may exploit richer spatio-temporal models. For instance one may utilize multiple tree structures providing additional coverage of the edges in the original loopy graphical model. A similar idea was exploited by Sapp *et al.* [44] who decompose the intractable model into a tree ensemble with the full coverage of the edge relationships of the original model. We do not know how the above improvements might increase the accuracy of the algorithm, and we leave it as an interesting problem for future research.

Figure 5.2: Examples of pose estimates of the detection approach. The results are compared to the results of FMP [55]. The first and third rows contain the results of FMP, the second and fourth rows contain the results of our method on the same images.

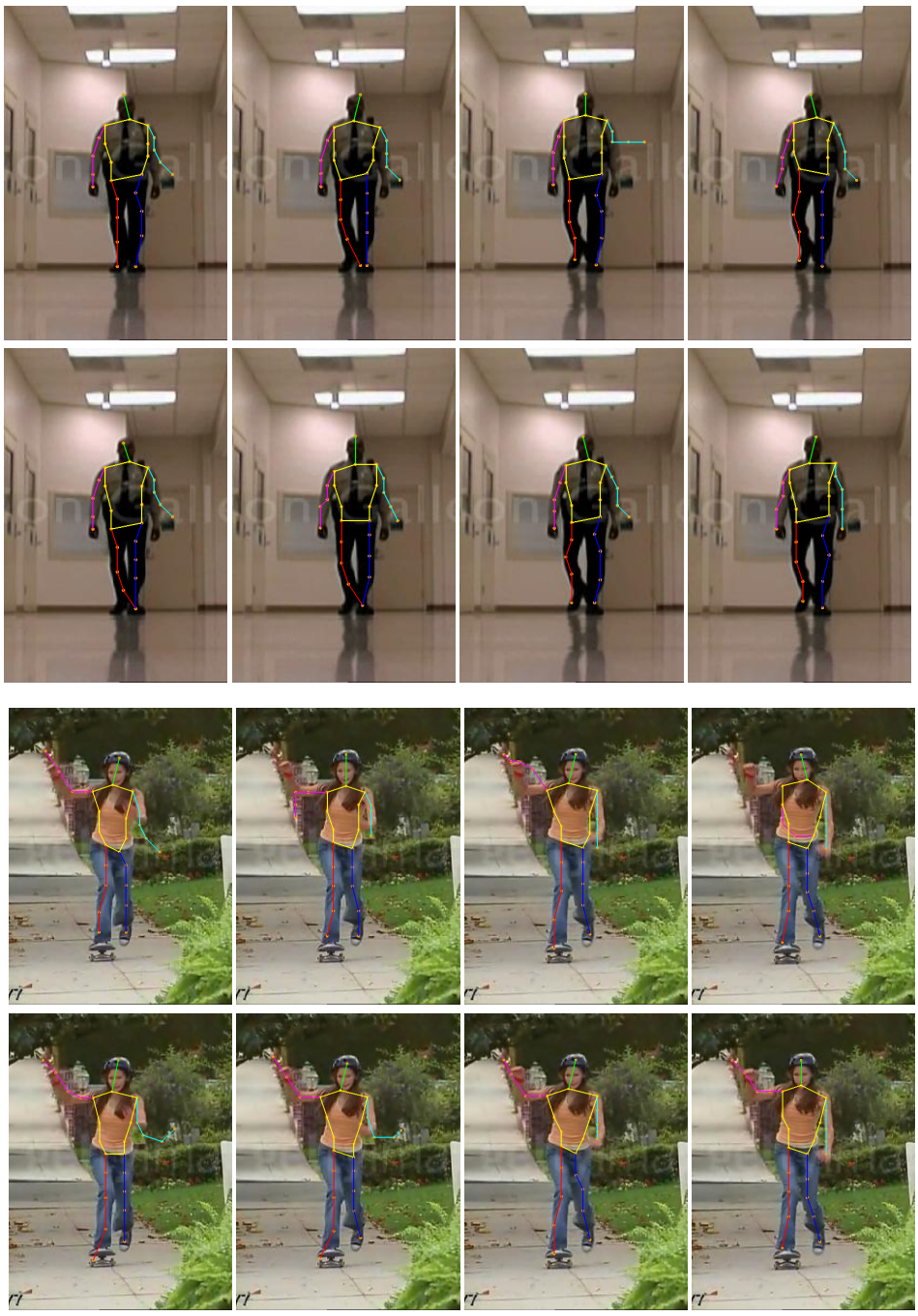


Figure 5.3: Examples of pose estimates of the detection approach. The results are compared to the results of FMP [55]. The first and third rows contain the results of FMP, the second and fourth rows contain the results of our method on the same images.

Chapter 6

Abstracting Human Pose Estimation

Human pose estimation is a challenging problem and an active research field, motivated by many applications of pose detection, such as human-computer interaction, surveillance and gaming. Recent advancements in pose estimation [46] are able to give results sufficiently good for the use in industrial and/or commercial applications, such as Microsoft Kinect^{TM}. However, using a state-of-the-art algorithms in real-world applications has numerous challenges. The majority of software engineers are non-experts in Computer Vision and pose estimation and they may encounter many problems, including the following:

- The state of the art advances fast and it is hard to track it down, as there is no regularly updated list of benchmarked and evaluated pose estimation algorithms. Furthermore, there is no one method that would work best in all circumstances.
- Given a state-of-the-art pose estimation academic paper, it is not trivial for non-experts to implement it. Furthermore, it is very hard to constantly reimplement the pose estimation algorithm for a specific application to keep up with the state of the art.
- The interface to most ready-to-use algorithms requires understanding of the parameters.

We believe that the solution to these problems may be addressed with the notion of the task, which we define as a combination of input description and output requirement, as well as parameters that can affect the result. This provides enough information to select the appropriate algorithm, while hiding the implementation details behind the abstraction. If the abstraction covers enough of the problem space, new algorithms can be seamlessly integrated without any changes to the interface, which would provide users with continuous updates to the state of the art. Furthermore, the requirements

of a specific platform may be taken into account, e.g. by utilizing low-power algorithms for mobile devices.

The purpose of this chapter is to introduce a task-based human pose estimation control system. We focus on the problem of 2D pose estimation in our selection of algorithms used in the system. However, the design of the abstraction is sufficiently general to accept other types of algorithms, such as 3D pose estimation and pose estimation in stereo. The system was designed as part of OpenVL [36], a framework that abstracts some Computer Vision problems such as segmentation, matching and image registration.

The two key contributions of this chapter are:

- A task-based abstraction for human pose estimation, which hides implementations of various pose estimation algorithms behind a single simple yet powerful application programming interface (API)
- A method for mapping from task to algorithm that automatically selects method most likely to succeed and adjusts its parameters based on the task description

This chapter is organized as follows. We first discuss task description in Section 6.1, and then outline the mapping from the task description to the algorithm which produces final pose estimates. Section 6.3 present the experimental evaluation of the algorithm mapping, Section 6.4 is devoted to discussion and future work.

6.1 Task Description

The task description is based on three categories: input, output, and target.

6.1.1 Input Description

Input type. In our definition of abstraction we would like to capture as many combinations of input data as possible. With this in mind, we format the input data as a temporal sequence of spatial arrangements. Every spatial arrangement is determined by a set of images coupled with poses of the corresponding cameras at the current moment. The poses may be undefined, while images contain information about color, depth or both. Temporal sequences may also contain data such as whether the video sequence is being streamed or is available at once. We refer to this as the input type.

The above definition of Input Type is flexible enough to cover most of the common combinations of cameras in time and space. For instance, it

naturally represents the setting of multiple calibrated cameras, where each spatial arrangement captures the position of the cameras at every time instance. The stereo vision system may be described by spatial arrangements, each containing two color images. The setup when the single camera is moving with an unknown trajectory is described by spatial arrangements with single image and undefined camera position each. The common case of a single depth camera is handled by one spatial arrangement with single depth image.

Image description. In addition to input type we include an image description, which encompasses the user's prior knowledge about the input image data. We define two types of image description: amount of *occlusion* and *clutter*. We think they are the most relevant to the general description of an image or video sequence in the setting of 2D pose estimation. However, in other cases such as ones utilizing depth images there might be different relevant conditions, which may be added as part of future work.

We loosely define the *clutter* as how many features would likely be found in regions of the image not belonging to a person. For instance, an image with a person standing on a field would possess low clutter, while an image of a person in a city setting with many cars and buildings in the background would be considered as high clutter. The *occlusion* condition reflects how likely human bodies or their parts are to be covered by elements in the scene, such as a person standing behind of a desk. Both clutter and occlusion are defined in the range of $(0, 1)$ and we quantize them into *Low, Medium, High*, as most of the problems do not require an in-depth description.

Our input description consists of the input type and image description as defined above, which we use to select the algorithm most likely to succeed in the current case. The input type constrains the set of algorithms our framework may select, because most of the algorithms strictly define what kind of data they work on, such as color images or or stereo vision pairs. We use image description as a factor to find the best algorithm for the given input data.

6.1.2 Output Requirement

Similarly to input description, we would like the output requirement to cover most of the common body representations that one might want to infer from the image or video data. With this in mind we first define a set of all body parts that we include in the framework: *Head, Neck, Chest, L/R Shoulder, L/R UpperArm, L/R Elbow, L/R LowerArm, L/R Hand, Abdomen, L/R Hip, L/R UpperLeg, L/R Knee, L/R LowerLeg, L/R Foot.* Although finer-

grained representations such as one including fingers may fit well into our framework, we are leaving this as a future work. We further include body part composition, which is a set of body parts with a description of the requirement regarding the included parts. The composition requirement may include one or several of the following:

- 2D or 3D location relative to camera
- Orientation as roll, pitch and yaw relative to camera
- Pixel-wise mask
- Bounding box or cube

Finally, we define the output requirement for the task as a set of body part compositions. We also predefine a set of common compositions including *Full-Body, Upper-Body, Head+Torso, Head.* Note that if the system detects more than one person, it returns the required information about each of them. We also include the speed/accuracy requirement in the range of $(0, 1)$, determining how much the accuracy could be sacrificed for the speed.

The above representation captures the results one may get from the majority of pose estimation algorithms. For instance, kinect-style body part labelings together with 3D positions of body joints may be described by a set of compositions, each of which requires a 3D location of the joint and a pixel-wise mask. A set of 2D body joint positions together with the person's silhouette may be represented as a set of single-part compositions with the requirement of a 2D position, together with a composition of all body parts, requiring a pixel-wise mask. A simple face detector may be described as a single composition with a single body part *Head* with the only requirement of the bounding box.

Note that the above description may encompass a set of other algorithms in Computer Vision, generally not associated with pose estimation such as face or head detection and pose orientation estimation. However, we believe that they share common features and algorithms with the field of pose estimation and person detection and should be considered together (see Section 2.1.1).

6.1.3 Target Description

The target description allows users to encode their prior knowledge about people in the image, which consists of one or more of the following:

- The population
- The set of compositions with defined priors that include visibility, location, size and orientation in 2D or 3D
- The distinctiveness from background in terms of color, texture, or motion, in the range $(0, 1)$

We define the population as the number of people in each image. The size and location of body part compositions are defined in the range $(0, 1)$ relatively to the size of the input image. The above conditions may affect the selection of the algorithm. For example, there may exist algorithms that specifically target multiple-people scenarios or work well on low-resolution images. Conversely, there may be algorithms that fail on images when certain body parts are invisible, e.g. lower body. The distinctiveness from background may be given when the user knows something about the appearance or motion of the person in the video. High color distinctiveness may favour methods that rely on color while high texture may be important for certain gradient-based algorithms. High motion distinctiveness tells the system that the person is moving fast compared to the background, and certain methods involving motion-based segmentation may come into play. Alternatively, certain pre-processing based on color, gradient or motion features may be applied.

Furthermore, the user may also have prior knowledge of the person's pose, such as visibility, location, size or orientation of certain body parts, which may play a role in the algorithm selection process. For instance, prior knowledge of legs being hidden behind the desk may trigger the selection of an upper-body pose estimation algorithm, or the prior that the person is facing the camera may help select the face orientation method instead of head orientation estimation algorithm. The prior knowledge of the above conditions may be utilized by algorithms that employ instance-specific learning applied on top of reliable detections of canonical poses [41]. Alternatively, this may be directly incorporated by certain algorithms. For instance, local features involving invisible body parts may be weighted low in the pose estimation procedure while a location prior may increase scores for certain parts in the image. Prior knowledge of size of body parts may weight certain scales higher, and orientation prior may be used to increase weight for orientation-specific features in an algorithm.

In addition, we provide several pre-defined body poses that may be used instead of body compositions: *Regular*, *Unusual* and *Front-Facing*. The *Unusual* pose is a non-vertical or highly articulated body configuration and

(a) (b) (c) (d) (e)

Figure 6.1: Examples of scene conditions and algorithms output. The results from three algorithms are presented, from top to bottom: GBM [43], F-FMP [55], U-FMP [55]. (a) Regular Pose (b) Unusual Pose (c) Lower Body is invisible (d) High Clutter (e) Low Clutter and Large Size. These algorithms are described in Section 6.2.

Regular is any other pose. *Front-Facing* is a pose when torso of the person is roughly facing the camera. See Figure 6.1 for some examples of images with a defined image description or pose prior together with the outputs from the algorithms that we include in the framework.

6.2 Task to Algorithm Mapping

Based on the abstraction outlined in the previous section we now present a proof-of-concept framework designed to demonstrate the utilization of the abstraction. In this work we consider four algorithms for 2D body pose estimation: Rothrock's grammar-based model (GBM) [43], Flexible Mixture of Parts [55] for upper body (U-FMP) and full body (F-FMP) and our shortest path approach from Chapter 5 (SPA). Furthermore, we include two algorithms for head and face orientation estimation in order to demonstrate the utility of the part-wise requirement formulation in the abstraction: Face orientation estimation by Zhu and Ramanan [57] (FO) and head/torso orientation prediction algorithm by Maji *et al.* [34] (HTO). We selected the

Table 6.1: Abstraction condition matrix. The task controls are presented in the first two columns, followed by the algorithms used in our proof-of-concept abstraction. The level of satisfaction for each control per algorithm forms the basis for algorithm selection based on the user-supplied description. Note: FB = Full Body, UB = Upper Body, LB= Lower Body, FF=Front-Facing, H+T=Head+Torso; L=Low, M=Medium, H=High; px = pixels.

	Controls	GBM [43]	F-FMP [55]	U-FMP [55]	SPA	HTO [34]	FO [57]
Input Type:	Image	✓	✓	✓	✗	✓	✓
	Video	✓	✓	✓	✓	✓	✓
Image Description:	Clutter	L	M-H	M-H	M-H	L-H	L-H
	Occlusion	L	M-H	M-H	M-H	H	L-M
Target Description:	Population	1	>= 1	>= 1	1	>= 1	>= 1
	Size (px)	$80-300$	$50-500$	> 300	$50-500$	> 20	> 80
	Pose	All	Regular	FF	Regular	All	FF
	Invisibility	✗	LB	LB	LB	✗	✗
Output Requirements:	Joint Locations	FB,UB	FB,UB	UB	FB,UB	✗	Head
	Joint Orientation	✗	✗	✗	✗	H+T	Head
	Accuracy/Speed	✓	✗	✗	✓	✓	✓

algorithms for the framework based on the problem space coverage, performance and code availability on the web. Methods for pose estimation return 2D locations of body joints, while head/face orientation estimation algorithms return the yaw angle in degrees. All methods operate on color images or image sequence in the case of SPA.

6.2.1 Algorithm Selection

We use the task description presented from the previous section to select the appropriate algorithm. Table 6.1 shows the conditions matrix that we use for the algorithm selection. The Input Type row reflects that only the SPA algorithm does not work on single images. Image description specifies the level of occlusion and clutter different algorithms can tolerate.

Target description identifies that only GBM and SPA a require single person in the image. Size describes on which pixel sizes of relevant compositions the algorithm work best. For pose estimation algorithms GBM, F-FMP, U-FMP and SPA the size of the body is reflected in the table, while

we use size of the face or head for HTO and FO. Pose specifies the poses for which the algorithm has relatively high performance. GMB works relatively well on all poses, F-FMP and SPA require a roughly vertical pose and U-FMP works best when the person is facing the camera. Likewise, FO works on front-facing people, while HTO works in all circumstances. Invisibility reflects that GBM does not work as well as FMP when parts of the body are not visible.

Output requirements specify what kind of output each algorithm is able to produce. We can see that among pose estimation algorithms only U-FMP cannot return full body pose. In contrast to FO, HTO is not able to give any information about the location of the head. Furthermore, among all algorithms only HTO and FO can return orientation of body parts, Head+Torso and Head correspondingly.

In order to select the appropriate algorithm for a specific task, the system performs the following steps:

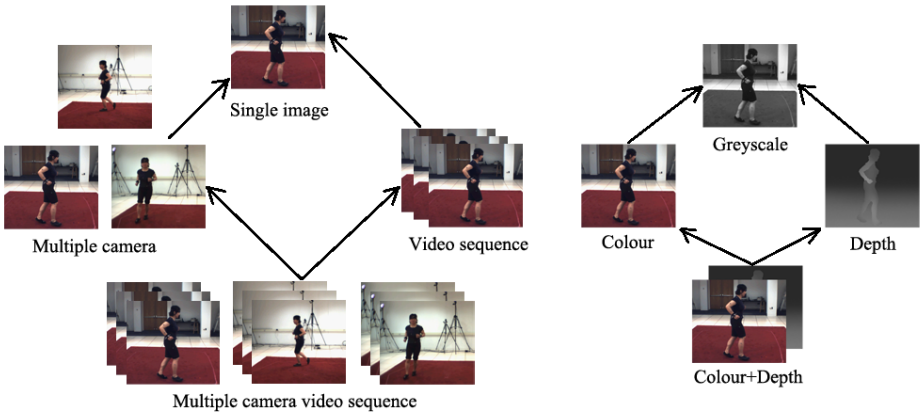
1. Searches the task conditions matrix (Table 1) for all methods that would satisfy the input description and output requirements. If no algorithm covers the provided specification, the closest algorithm is chosen (see Section 6.2.2).

2. From the chosen algorithms it selects the ones that satisfy the conditions of the target description sequentially for each condition in the following order: support for multiple people; support for invisibility, size, orientation and location priors; support for color, texture or motion priors.

3. Chooses the fastest algorithm among the ones obtained in the previous step and adjusts its parameters according to the speed/accuracy requirement.

6.2.2 Closest Algorithm Search

The input type of a task is either a single image, a temporal sequence of frames, multiple images from calibrated cameras or multiple video sequences from calibrated cameras. This could be represented by a directed graph, such that every edge from a parent to its child induces loss of information, such as conversion from video sequence to a single image by discarding all frames but the current one (Figure 6.2 (a)). Furthermore, every image or video frame contains either grayscale, color, depth or both color and depth information, which may be likewise described by a directed graph (Figure



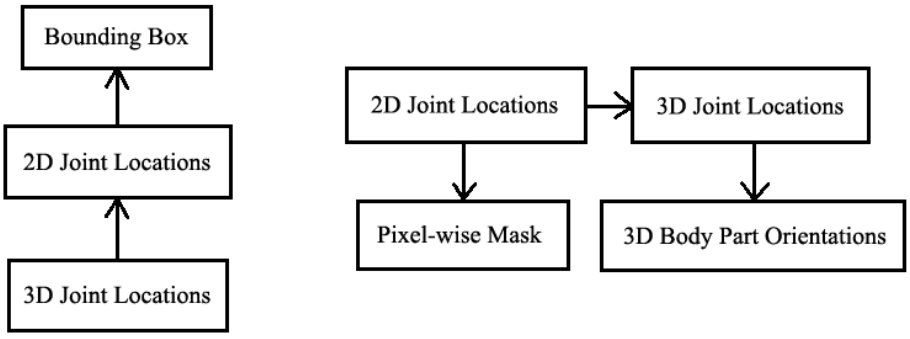
(a) (b)

Figure 6.2: Graph representing input types. (a) Input types form a 4-node graph. (b) Image types form a 4-node graph.

6.2 (b)). The final definition of input type includes both aspects outlined above. Therefore, the set of all possible input types may be represented by a directed graph containing 4×4 nodes, where the less informative type is a single grayscale image, while the most informative contains temporal video sequences of color images with depth from multiple cameras.

Our output requirements are represented by two graphs G_O^L and G_O^D, where each edge identifies a data conversion process. Edges in graph G_O^L involve loss of information (Figure 6.3), while edges G_O^D involve deriving the non-existent information using certain assumptions or default values (Figure 6.3). For instance, 2D joint locations are obtained from 3D joint locations by projecting them onto the image plane, and pixel-wise body part labeling may be obtained from the positions of 2D body joints using a simple puppet model, whose size is determined by the average limb length in the skeleton.

Suppose the task description includes input data I and output requirement O. If no algorithm in the framework fits the provided task description, the procedure described in Algorithm 2 is performed with G_O^L replacing G_O. The algorithm returned by the solution works on at least one of the sub-sets of the given input data and returns the results, at least as detailed as the requirement is. In this case our framework directly supports the given task description, and if the procedure returned more than one algorithm, further selection takes place as outlined in the previous section. If the procedure



(a) (b)

Figure 6.3: Graphs representing output requirements. (a) Graph G_O^L reflects loss of information in the data conversion process. (b) Graph G_O^D identifies deriving additional data during the conversion process.

does not find any algorithm, the framework repeats it, but uses G_O^D instead of G_O. In contrast to the previous attempt, the returned algorithm infers certain parts of the data, and it is labeled as *Inferred*. For example, the algorithm may return 2D joint positions, but the conversion procedure will assume the Z axes to be 1 for all joints. The latter procedure will always return an algorithm, because the bottom of the chain is a $2D$ pose estimation algorithm included in the framework that works on grayscale images, which will be selected in the worst case.

Input: Input type I in a graph G_I, output requirement O in a graph G_O
Output: Pose estimation algorithm
Algorithm $\leftarrow \emptyset$;
Queue $\leftarrow \emptyset$; Queue $\hookleftarrow O$;
for *Queue* $\neq \emptyset$ **do**
  $\hat{O} \hookleftarrow$ Queue; Queue $\hookleftarrow$ *parents*$(\hat{O})$
  Queue$_2 \leftarrow \emptyset$; Queue$_2 \hookleftarrow I$;
  for *Queue*$_2 \neq \emptyset$ **do**
    $\hat{I} \hookleftarrow$ Queue$_2$; Queue$_2 \hookleftarrow$ *children*$(\hat{I})$
    if *Exist algorithm A for* $\hat{I}$ *and* $\hat{O}$ **then**
      Algorithm $\leftarrow A$;
      exit;
    end
  end
end

Algorithm 2: Closest algorithm search. The procedure finds the closest algorithm that matches input type I and output requirement O, assuming input type graph G_I and output requirement graph G_O. By $\hookleftarrow$ we denote the operation of taking and putting an element into a set.

6.2.3 Parameter Derivation

The task description is also used to derive the appropriate parameters for the chosen algorithm. Often many of the parameters of an algorithm are learned and included with the model. However, usually there are certain parameters one has to tune according to one's needs. In our current set of algorithms we only adjust parameters that affect the speed/accuracy tradeoff. As can be seen in Table 6.1 all algorithms but FMP can be tuned in accordance with the user's requirement for speed. GBM requires scale parameters to be specified, which are set based on the prior knowledge of target size and speed constraints. SPA has controls that specify likelihood of guessing the correct pose and the amount of smoothing, which is tuned for the required level of speed/accuracy. FO comes with three pre-trained models of different levels of detail and different inference speeds, which are set automatically. HTO directly provides the parameter that affects speed and accuracy.

6.3 Algorithm Selection Evaluation

Every algorithm has a concrete input type that it can accept and output requirements it is able to satisfy, which allows us to fill Input Type and Output Requirements rows of the condition matrix 6.1. However, filling the Image Description and Target Description rows requires an insight into the performance of the algorithms under various task conditions, which we determine with the help of experiments. We selected 120 images from five pose estimation datasets: Buffy Stickmen [16], Image Parse [40], Leeds Pose Dataset [26] and Synchronic Activities Stickmen [12]. We selected the images based on the maximum coverage of the task description problem space, and manually annotated them with the following labels:

- The amount of clutter
- The amount of occlusion
- Lower body visibility flag
- Target size
- Pose label

We measure clutter as the distinctiveness from the background in terms of occlusion and clutter. Pose is labeled to be either *Regular* or *Unusual* and may be *Front-Facing* (in the abstraction, *All* is equal to *Regular+Unusual*). Non-vertical or highly articulated body configurations were labeled as *Unusual*, roughly facing the camera as *Front-Facing*, all others as *Regular*. Furthermore, we cropped the images such that they each contained a single person only in order to maintain consistency, as GBM works only on single person images.

We ran GBM, and F-FMP and U-FMP on 70 images out of 120 selected and filled the task conditions matrix based on obtained evidence of performance of the algorithms, leaving the remaining 50 images for testing. We found that F-FMP is the preferable algorithm for the task of full-body pose estimation in the presence of clutter, as GBM is more likely to pick suitable clutter as a body part. At the same time GBM works better in the absence of clutter, while FMP is more likely to miss a body part in an uncluttered environment. We think this can be explained by the fact that GBM utilizes segmentation to better distinguish foreground from the background. Furthermore, F-FMP produces limb double counting more frequently than GBM, which may be explained by the fact that it is a tree model which

does not share knowledge between its limbs. Furthermore, F-FMP produces subtly less accurate results in the case of non-vertical or highly articulated poses, as it seems to have stronger priors towards such poses in its training dataset. We use SPA in the cases when the video sequence is available.

For upper body pose estimation we found that U-FMP is the preferred method when the image is sufficiently large or the person is facing the camera. It also works better than other algorithms when the prior knowledge of lower body being occluded is given, although F-FMP is more likely than GBM to correctly detect the hands of the person even when there is no evidence of legs in the image. For the task of head yaw estimation we found that the main control that affects the selection between HTO and FO is the prior knowledge of the size of the head or orientation.

In order to evaluate the task-based algorithm mapping, we ran three pose estimators GBM, F-FMP and U-FMP on the remaining 50 images and determined the system success rate, which was defined by how often the best algorithm for a specific task is selected. The success rate was 76%, and in cases where it picked a non-optimal method the difference in accuracy between the best and selected algorithms was approximately 15%, therefore the result would be relatively close to optimal. This supports our insights on the performance of the algorithms and shows that the system selects pose estimation algorithms reasonably well. Figures 6.4-6.6 illustrate some examples of the results of our algorithm selection process. Every row corresponds to a single algorithm, while every column represent an image with a task description. The algorithm selected by our system is marked in green, the correct one is marked in blue.

6.4 Discussion and Future Work

This chapter is intended to demonstrate that a wide-coverage abstraction over human pose estimation targeted at non-expert users is possible, and a task-based approach provides a reasonable level of control while still hiding the complexity of algorithmic details and parameters under a single flexible application programming interface. Our abstraction utilizes various kinds of input data together with image description, target prior knowledge definition and output requirements to cover a large volume of the pose detection problem space. The condition matrix together with closest algorithm search procedure 2 maps the task description to a suitable pose estimation algorithm and automatically derives the necessary parameters. Our results demonstrate the advantages of the current approach.

(a) (b) (c)

Figure 6.4: Algorithm mapping evaluation for full-body pose estimation. Images with various descriptions are taken: (a) Regular Pose, High Clutter, Low Occlusion, Small Size; (b) Regular Pose, Medium Clutter, Low Occlusion, Large Size; (c) Unusual Pose, High Clutter, Low Occlusion, Small Size. Algorithm, selected by the abstraction is marked in green, the correct one is marked in blue. Algorithms from top to bottom: GBM [43], F-FMP [55]. Best viewed in color.

The main flaw of the system in its current state is the fact that it requires expert knowledge to be encoded into the condition matrix. If one wants to add an algorithm into the current framework, one would have to run it on the same set of images we used to test other algorithms and expand the condition matrix in accordance with the results. However, there is no guarantee that the parameters that we chose in our framework such as clutter and occlusion give the best prediction for the efficiency of the algorithms.

The solution to the above might be provided by an automated run-time algorithm selection process based on the features extracted from input images or video sequences. This may be guided by a multi-class classification procedure, where every class corresponds to a certain algorithm that fits the given task conditions. During inference the system would select the best algorithm according to the output of the classifier. The classifier may be trained on the features extracted from a set of training images with anno-

(a) (b) (c)

Figure 6.5: Algorithm mapping evaluation for head yaw estimation. Images with various descriptions are taken: (a) Large Size, Non-Front-Facing; (b) Small Size, Front-Facing; (c) Large Size. Algorithm, selected by the abstraction is marked in green, the correct one is marked in blue. Algorithms from top to bottom: HTO [34], FO [57]. Empty annotation reflect algorithm failure. Best viewed in color.

tated poses, which would be coupled with class labels, specifying the best-performing pose estimation algorithm. In our opinion a broad set of various features may be beneficial. Together with a weighting classifier such as SVM one may find out which features have the most influence in algorithm selection process. Furthermore, in practice feature computation should not be time-consuming, so features like BRIEF [7] or Haar-like features [50] may come into play. With this setup the new algorithms can be seamlessly added into the framework, without any prior expert knowledge about their performance, as the training process outlined above would automatically re-train the classifier. Furthermore, the set of spatial features may be changed or expanded at any time with the same effect. Note that similar approaches have been tried in other areas in Computer Science, utilizing machine learning for aiding algorithm selection process [23] and runtime prediction [24]. In addition, one may explore the ways of automatic adjustment of the parameters for each algorithm in the system. We leave such an automated classification procedure to future work.

Figure 6.6: Algorithm mapping evaluation for upper-body pose estimation. Images with various descriptions are taken: (a) Medium Clutter, Low Occlusion, Small Size; (b) Low Clutter, Low Occlusion, Small Size; (c) Low Clutter, Large Size, Lower Body Invisible. Algorithm, selected by the abstraction is marked in green, the correct one is marked in blue. Algorithms from top to bottom: GBM [43], F-FMP [55], U-FMP [55]. Best viewed in color.

Chapter 7

Conclusion

In this thesis we consider the problem of human pose estimation. We present two novel algorithms for monocular 2D pose estimation from video sequences. The first one aggregates the information from adjacent frames and then searches for a shortest path of pose estimates from the output of a single-image pose estimator throughout the whole video sequence, significantly outperforming the state of the art for single-image pose estimation. The second algorithm utilizes a spatio-temporal tree model and for every video frame performs articulated human detection, taking into account several previous frames, demonstrating state-of-the-art pose estimation performance.

Furthermore, we release the UCF Sports Pose dataset, which consists of full-body human pose annotations for a subset of videos from the UCF Sports Action dataset that contain people in roughly vertical positions. Furthermore, we propose a new metric for the evaluation of pose estimation results, which better reflects the performance of the current state of the art algorithms for 2D human pose estimation. In addition, we release a highly configurable Video Pose Annotation tool that greatly simplifies the manual process of annotating poses in video sequences.

Finally, we present a novel abstraction over human pose estimation that captures a large volume of the pose estimation problem space. The abstraction comes with a notion of a task description, which includes the description of the input data and the output requirements for the pose estimation problem. It also includes a meta-algorithm that maps a task description to a pose estimation algorithm that is expected to give the best results on the specific problem based on the expert knowledge about the algorithms in the framework.

Future work for each part of this thesis is discussed in detail in the end of each chapter. The future work on the first video pose estimation algorithm may focus on the utilization of a better and faster method of tracking the poses. In addition, learning of the patterns of temporal co-occurrence of appearance may be employed. An alternative research direction is to investigate the ways to make the algorithm work in an on-line setting. Future

work on the second algorithm encompasses joint spatio-temporal learning, improved tracking of poses and exploration of different tree models.

The abstraction of human pose estimation may benefit from utilization of a machine learning technique for the expert knowledge directly encoded into the system. The technique may work on image features and consider the task as a classification problem, where each class corresponds to a specific pose estimation algorithm. Furthermore, one may want to explore ways to automatically tune the parameters of each algorithm.

Bibliography

[1] S. Amin, M. Andriluka, R. Marcus, and B. Schiele. Multi-view Pictorial Structures for 3D Human Pose Estimation. In *British Machine Vision Conference*, 2013.

[2] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition*, 2009.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library.* 2008.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *IEEE European Conference on Computer Vision*, volume 3024, pages 25–36. 2004.

[5] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61:211–231, 2005.

[6] P. Buehler, M. Everingham, D.P. Huttenlocher, and A. Zisserman. Upper body detection and tracking in extended signing sequences. *International Journal of Computer Vision*, 95:180–197, 2011.

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *IEEE European Conference on Computer Vision*, pages 778–792, 2010.

[8] K. Chiu and R. Raskar. Computer vision on tap. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 31–38, 2009.

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

[10] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *British Machine Vision Conference*, 2009.

[11] M. Eichner and V. Ferrari. We are family: Joint pose estimation of multiple persons. In *IEEE European Conference on Computer Vision*, 2010.

[12] M. Eichner and V. Ferrari. Human pose co-estimation and applications. *IEEE Transasctions on Pattern Analysis and Machine Intelligence*, pages 2282–2288, 2012.

[13] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 617–624, 2011.

[14] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1627–1645, 2010.

[15] P.F. Felzenszwalb and D.P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.

[16] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[17] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *International Conference on Machine Learning*, pages 304–311, 2008.

[18] O. Firschein and T.M. Strat. *RADIUS: Image Understanding For Imagery Intelligence*. Morgan Kaufmann, 1st edition, 1997.

[19] K. Fragkiadaki, Han Hu, and Jianbo Shi. Pose from flow and flow from pose. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[20] P. Guan, A. Weiss, A.O. Balan, and M.J. Black. Estimating Human Shape and Pose from a Single Image In *IEEE International Conference on Computer Vision*, 2009

[21] K. Hara and R. Chellappa. Computationally efficient regression on a dependency graph for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013

[22] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[23] F. Hutter, D. Babi, H.H. Hoos, and Alan J. Hu. Boosting verification by automatic tuning of decision procedures. In *Formal Methods in Computer Aided Design*, pages 27–34, 2007.

[24] F. Hutter, Lin Xu, H.H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: The state of the art. *Artificial Intelligence Journal*, abs/1211.0906, 2012.

[25] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *IEEE International Conference on Computer Vision*, 2013.

[26] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *British Machine Vision Conference*, 2010.

[27] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *IEEE International Conference on Pattern Recognition*, pages 2756–2759, 2010.

[28] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, pages 995–1004, 2008.

[29] C. Kohl and J. Mundy. The development of the image understanding environment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 443–447, 1994.

[30] K. Konstantinides and J.R. Rasure. The khoros software development environment for image and signal processing. *IEEE Transactions on Image Processing*, 3:243–252, 1994.

[31] L. Ladický, P.H.S. Torr, and A Zisserman. Human pose estimation using a joint pixel-wise and part-wise formulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013

[32] C. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, 2009.

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

[34] S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3177–3184, 2011.

[35] T. Matsuyama and H. Vincent. Sigma: a framework for image understanding integration of bottom-up and top-down analyses. In *International Joint Conference on Artificial intelligence*, volume 2, pages 908–915, 1985.

[36] G. Miller and S. Fels. OpenVL: A task-based abstraction for developer-friendly computer vision. In *IEEE Winter Application and Computer Vision Conference*, pages 288–295, 2013.

[37] .B. Moeslund, A. Hilton, V. Krger, and L. Sigal, editors. *Visual Analysis of Humans - Looking at People*. 2011.

[38] G Panin. *Model-based Visual Tracking: the OpenTL Framework*. John Wiley and Sons, 2011.

[39] J. Peterson, P. Hudak, A. Reid, and G.D. Hager. Fvision: A declarative language for visual tracking. In *Third International Symposium on Practical Aspects of Declarative Languages*, pages 304–321, 2001.

[40] D. Ramanan. Learning to parse images of articulated bodies. *Advanced in Neural Information Processing Systems*, 2006.

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

[42] M.D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[43] B. Rothrock, S. Park, and S.-C. Zhu. Integrating grammar and segmentation for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[44] B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1281–1288, 2011.

[45] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *International Conference on Multimedia*, 2007.

[46] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[47] L. Sigal, A.O. Balan, and M.J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal on Computer Vision*, 87:4–27, 2010.

[48] E. Simo-Serra, A. Quattoni, C. Torras, and F. Moreno-Noguer. A Joint Model for 2D and 3D Pose Estimation from a Single Image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2013

[49] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2649, 2013.

[50] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518 vol.1, 2001.

[51] C. Wang, Y Wang, and A.L. Yuille. An approach to pose-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[52] Heng Wang, A. Klaser, C. Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3169–3176, 2011.

[53] J.M. Wang, D.J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:283–298, 2008.

[54] Shandong Wu, O. Oreifej, and M. Shah. Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories. In *IEEE International Conference on Computer Vision*, pages 1419–1426, 2011.

[55] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[56] T.-H. Yu, T.-K. Kim, and R. Cipolla. Unconstrained Monocular 3D Human Pose Estimation by Action Detection and Cross-modality Regression Forest. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[57] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2879–2886, 2012.

[58] S. Zuf, J. Romero, C. Schmid, and M.J. Black. Estimating human pose with flowing puppets. In *IEEE International Conference on Computer Vision*, 2013.