Dissertation

# Motion Kinematics and Dynamics Prediction Using Human Pose Estimation in Videos

Towards Automated, Kinematical Profiling
Of Swimmers and Ski Jumpers

Dan Zecha


Department of Computer Science
University of Augsburg

| | |
|---|---|
| Advisor: | Prof. Dr. Rainer Lienhart |
| Reviewers: | Prof. Dr. Rainer Lienhart |
| | Prof. Dr. Jörg Hähner |
| Thesis Defense: | July 10, 2019 |

# Abstract

The international success of world-class athletes depends strongly on the assessment and active improvement of their technique. Camera technology, force sensors, and sophisticated evaluation software have become common tools for compiling sophisticated biomechanical performance profiles, which allow for precisely evaluating the human motion and concluding necessary steps for improving the athletic performance. In this work, we investigate approaches and algorithms for the fully automatic, time-continuous estimation of kinematic and dynamic performance parameters of athletes in sports videos. Therefore, we focus on two specific application scenarios: estimating the motion kinematics of swimmers in a swimming channel and predicting the jump forces of ski jumpers just from video footage.

A fundamental concept of the kinematic analysis is the key-pose, a specific, well-defined athlete posture. The precise and continuous identification of key-poses allows for deriving a lot of relevant kinematic parameters: stroke frequency, kick rate, and inner-cyclic interval timings. We show that the identification of a key-pose can be treated as a classification problem and introduce an ensemble of spatiotemporal pose detectors for continuously estimating the occurrence of distinct postures in cyclic motion. While we can not guarantee that these postures represent actual key-poses, they can be used in a simple probabilistic scheme to indirectly infer key-pose occurrences. We furthermore discuss articulate pose estimation for identifying key-poses directly by their defining features. The proposed pose estimator leverages a deep neural network for learning athlete appearance, which enables the articulate estimation of joint locations even in the visually challenging scenario of a swimming channel. With modern pose estimation systems not being immune to estimation errors, we investigate the error susceptibility of two recent systems and derive a taxonomy of the most pertinent estimation errors. Several optimization strategies are discussed, where each method rectifies a different error class, consequently improving joint localization and key-pose identification.

At last, we address the question of how the jump forces of a ski jumper can be predicted from video footage. We introduce a neural network build on dilated convolutional layers for predicting a series of force measurements directly from a sequence of athlete poses. An experimental exploration of different architectures indicates the general feasibility of our approach.

# Danksagung

Ich bedanke mich herzlich bei Prof. Dr. Rainer Lienhart für die Betreuung und die Begutachtung meiner Arbeit, für viele kritische Diskussionen und einen regen Ideenaustausch in den vergangenen Jahren. Ich möchte mich außerdem bei Prof. Dr. Jörg Hähner für die Anfertigung des Zweitgutachtens bedanken.

Besonderer Dank gilt den Trainingswissenschaftlern, mit denen ich in den letzten Jahren zusammenarbeiten durfte und die mir durch viele Gespräche Einblicke in ihre Arbeit gewährt haben. Dr. Jürgen Küchler, Dr. Jens Graumnitz, Stephan Fuhrmann, Dr. Sören Müller und Dr. Sascha Kreibich haben mit ihren zur Verfügung gestellten Daten und ihrer Expertise direkt und indirekt zum Gelingen dieser Arbeit beigetragen.

Meinen Kollegen Christian Eggert, Moritz Einfalt, Philipp Harzig und Stephan Brehm danke ich für zahlreiche konstruktive Diskussionen, Denkanstöße und Ablenkungen während der letzten Jahre.

Ein besonderer Dank gilt meinem Vater, der mich stets unterstützt und ermutigt hat, meinen eigenen Weg zu gehen. Zuletzt möchte ich mich besonders bei meiner Frau Gloria bedanken, ohne deren Unterstützung und Ermutigung diese Arbeit nicht möglich gewesen wäre.

For my family.

# Contents


Abstract . . . . . iii
Acknowledgements . . . . . iv

**1 Introduction** **1**
1.1 Problem Definition and Challenges . . . . . 2
1.2 Contributions . . . . . 4
1.3 Publications . . . . . 6
1.4 Thesis Outline . . . . . 7

**2 Foundations and Datasets** **9**
2.1 Human Pose Estimation . . . . . 9
2.1.1 General Terminology . . . . . 9
2.1.2 Human Pose Graphs . . . . . 10
2.1.3 Human Pose Estimation in Monocular Images . . . . . 11
2.2 Kinematics and Dynamics in Videos . . . . . 12
2.2.1 Traditional Motion Kinematics . . . . . 13
2.2.2 Activity Kinematics . . . . . 14
2.2.3 Dynamics . . . . . 15
2.3 Datasets . . . . . 16
2.3.1 Swimming . . . . . 16
2.3.2 Ski Jumping . . . . . 23
2.4 Related Work . . . . . 25

**3 Activity Kinematics from Spatiotemporal Poselet Activation** **27**
3.1 Related Work . . . . . 28
3.2 Spatiotemporal Poselet Detection . . . . . 30
3.2.1 Poselet Training . . . . . 30

3.2.2 Temporal Poselet Clustering . . . 31
3.2.3 Classifier Training . . . 36
3.2.4 Spatiotemporal Poselet Mixtures . . . 37
3.2.5 Key-pose Estimation from Poselet Activation . . . 39
3.3 Experiments . . . 43
3.4 Summary . . . 50

**4 Athlete Pose Estimation with Deep Neural Networks 53**
4.1 Related Work . . . 54
4.2 Pose Estimation with Image Dependent Relation Terms . . . 55
4.2.1 Base Model . . . 55
4.2.2 Pictorial Structures as Computational Graphs . . . 59
4.3 Convolutional Pose Machines . . . 63
4.4 Key-pose Occurrences from Human Pose Estimates . . . 65
4.4.1 Pose Feature Series . . . 65
4.4.2 Random Forest Classification . . . 66
4.5 Experiments . . . 67
4.5.1 CG-IDPR Training . . . 67
4.5.2 Joint Localization . . . 68
4.5.3 Key-Pose Localization . . . 70
4.6 Summary . . . 72

**5 Kinematic Pose Rectification 73**
5.1 Related Work . . . 73
5.2 Pose Estimation Error Taxonomy . . . 74
5.2.1 Frequency . . . 77
5.2.2 Correlation . . . 78
5.2.3 Directional Offsets . . . 79
5.3 Joint Refinement . . . 80
5.3.1 Swap Optimization . . . 80
5.3.2 Directional Offset Minimization . . . 87
5.3.3 Temporal Robust Regression . . . 88
5.4 Experiments . . . 91
5.4.1 Joint Localization . . . 91
5.4.2 Key-Pose Retrieval . . . 93

5.5 Summary . . . 96

**6 Jump Dynamics Prediction of Ski Jumpers 97**
6.1 Related Work . . . 98
6.2 Multimodal Registration and Synchronization . . . 100
6.2.1 Measurement Setup . . . 100
6.2.2 Pose Estimation for Robust Camera Calibration . . . 101
6.2.3 Synchronization of Force and Camera . . . 103
6.3 Temporal Convolutional Model . . . 104
6.3.1 Dilated Convolution Blocks . . . 104
6.3.2 Network Structure and Losses . . . 106
6.4 Experiments . . . 108
6.4.1 Dataset . . . 108
6.4.2 Error Measures . . . 109
6.4.3 Tested Network Structures and Training Parameters . . . 109
6.4.4 Results . . . 110
6.5 Summary . . . 114

**7 Conclusion 117**
7.1 Summary . . . 117
7.2 Conclusion and Outlook . . . 118

**List of Figures 121**

**List of Tables 123**

**Bibliography 125**

# 1 — Introduction

Termed by the greek "$\kappa\iota\nu\eta\mu\alpha$" (kinema, engl.: movement, motion), "kinematics is the study of motion" [8] of particles or rigid bodies, without considering the causes of that motion. In classical mechanics, the study of particle kinematics involves arguing about the position of a particle, the displacement in position, commonly referred to as the trajectory, and quantities like speed, velocity, and acceleration. All these terms are equivalently used in the field of biomechanics, the "study of the movement of living things using the science of mechanics" [50]. The trajectories of a human's joints, their velocity, and acceleration are classical *kinematic parameters* describing the *motion kinematics* of a person. In sports, these classical parameters constitute the foundation to other kinematic characteristics. For example, the step count and length of runners, the stroke frequency of swimmers or the posture and flight trajectory of ski jumpers are all types of *kinematic parameters* that can be considered in the biomechanical profiling of an athlete.

The field of *dynamics*[1] studies forces and their association with motion, which includes forces that cause motion and vice versa. In a biomechanical context and especially in the biomechanical analysis of an athlete's *motion dynamics*, *dynamic parameters* include the pull forces of rowers or the jump forces of ski jumpers on a jump, swimmers on a starting block or high divers on a high diving platform, amongst others. Jump forces are measured using specialized equipment called *force plates* that can register and precisely determine the jump force of a person.

Adding to force measurement hardware, technology like 2d and 3d camera systems, inertial sensors and sophisticated evaluation software have become common tools in the daily routine professional athletes, their coaches and training scientists in general. As the international success of world-class athletes depends quite strongly

[1] Nowadays, the term *"dynamics"* is often used synonymously to the older term *"kinetics"*. When arguing about forces and their influence on motion, we will predominantly stick to the term "dynamics" in this thesis.

on the assessment and active improvement of his or her technique, technology is used to measure kinematic and dynamic parameters as a foundation of a sophisticated performance profile. Measurements are taken either directly via sensors or indirectly, for instance by measuring and evaluating motion kinematics via *manual annotation* of joints in video footage. Realistically, coaches usually consider only a qualitative evaluation, because almost all types of quantitative assessment of kinematic parameters are immensely time-consuming and therefore typically reserved only for the top athletes. This raises the question of how to exploit and reasonably evaluate the enormous amounts of data that camera systems produce without having to spend an unreasonable amount of time and energy.

Recent breakthroughs in the field of computer vision and deep learning have made the problem of automatically evaluating large sets of video data more approachable. A fully automatic analysis that captures motion kinematics and derives all types of kinematic parameters for an elaborate biomechanical analysis creates new possibilities in the broad field of performance diagnostics.

*This thesis investigates algorithms and approaches for the estimation of kinematic and dynamic parameters in sports videos with a specific focus on swimming and ski jumping.*

The goal is to use computer vision algorithms to get one step closer to a fully automatic, time-continuous performance analysis of athletes in training footage.

## 1.1 Problem Definition and Challenges

In the field of competitive swimming, a quantitative evaluation of an athlete's motion is highly desirable to supplement the typical qualitative analysis. We consider the following real-life application of a camera system for evaluating the technique of swimmers. An athlete swims in a swimming channel, a small pool where the water is accelerated to constantly flow from one end of the pool to the other. The basin has at least one glass wall and is monitored with multiple cameras from different angles. The athlete then dives into the current and starts performing swimming strokes in one of the four major swimming styles: butterfly, backstroke, breaststroke, and

freestyle. For the time period of the test, he or she is filmed by all cameras. The video footage constitutes the essential basis for the assessment of cyclic motion.

From the perspective of a human expert like a coach or training scientist, the evaluation timeline for the video footage is arduous. In a typical evaluation scenario, video footage is collected from a whole team of top-athletes and multiple, controlled swim tests. Typical evaluation protocols include a quantitative and qualitative analysis. The qualitative evaluation is determined by simply looking at the footage and using expert knowledge to rate the posture of the athlete and infer clues on how to optimize the motion sequence. A quantitative analysis is associated with a lot of manual labor. The expert has to count strokes and kicks, measure stroke times and, even more intricate, determine inner-cyclic motion intervals and their execution times. This involves clicking through a lot of images - frame rates of 50-60 Hz are common - in order to precisely identify athlete postures (dubbed *key-poses*) that constitute motion interval boundaries. The time that passes between those boundaries is evaluated over multiple strokes and offers valuable answers to the question of how to optimize even the last grain in an otherwise highly competitive swimming motion.

Automating the delineated evaluation protocols with computer vision algorithms involves several steps, some of which are trivial for a human, but difficult to express and reliably solve with algorithms. Three main problems have to be addressed: Detecting and tracking the athlete, estimating some form of pose representation and inferring kinematic and possibly dynamic parameters.

The **detection and tracking** of an athlete in all frames of a video serves mainly two purposes. Firstly, it is beneficial for a pose estimation algorithm to only evaluate parts of the image where a person is actually depicted. Hence, a person detector naturally reduces the size of the hypothesis space of pose locations, thereby increasing the pose estimation precision by excluding potential false positive image regions while lowering computational costs. Secondly, the continuous tracking of athletes itself can be used for kinematic evaluation. Consider the sports of ski jumping, where tracking allows for predicting the flight trajectory and velocity of the athlete, both of which are important indicators for judging a jump.

Naturally, the question of determining performance parameters of athletes is inherently linked to the human body and thereby **human pose estimation**, as motion kinematics are usually defined on the position, velocity, and acceleration of

joints and body parts. Measuring performance parameters involves detecting relevant parts of the pose or - ideally - estimating the pose directly by detecting the locations of major joints. Using the example of visually challenging swimming footage, direct pose estimation has been a difficult problem with unsatisfactory results for the longest time. With the recent developments in feature learning techniques via deep convolutional neural networks, finding good estimates for the complete skeletal pose is not only feasible but also yields decent results. Emerging pose estimation errors weaken the final pose estimation and therefore have to be addressed.

Depending on a specific parameter type, the **deviation of biomechanical motion quantities** like kinematics and dynamics can be very simple. Automatically counting strokes and leg kicks is a trivial task given that we are able to reasonably predict the human pose. The task of identifying the boundaries of inner-cyclic intervals is akin to the task of activity recognition in computer vision. For instance, the activity "doing push-ups" can be partitioned into two phases, denoted *sub-activities*, dropping to the ground from an outstretched posture and pushing up again. In the context of biomechanical analysis, meticulously identifying distinct motion intervals by standards of training scientists who emphasize utterly precise measurements proves to be a big challenge.

## 1.2 Contributions

The contributions of this thesis can be summarized as follows:

- **Spatiotemporal pose detectors**: The identification of specific human poses in aquatic environments is a challenging problem due to various forms of noise, refraction, occlusion, and foreshortening. Presuming that evaluation-critical key-poses can't be identified directly and are therefore *latent* or "hidden", we propose an identification scheme by means of a *spatiotemporal pose detector*. The detector is built from an ensemble of classifiers called poselets, each of which acts as a detector for the presence or absence of specific body part configurations in images. The whole ensemble continuously and reliably detects specific body postures, which we denote *support-poses*. We use these specific pose occurrences in a simple probabilistic scheme to infer the occurrence of latent key-poses.

- **A pose estimation network with image dependent pairwise relations**: We develop a deep neural network for human pose estimation of athletes. The network consists of two parts. The first part is a convolutional neural network that learns to identify the appearance of an athlete's joints directly from image data. The second part puts together possible joint detections in an image to form a complete and reasonable pose estimate. The formulation of the pose estimation network includes the definition of an operation called *max-convolution*, which allows for expressing deformation between two joints, i.e., the possibility that two neighboring joints may appear in a slightly deformable configuration. The proposed network is furthermore augmented with *image dependent pairwise relations*, additional terms that use the image content around a joint to help to predict the positions of adjacent joints. The athlete poses estimated with the network are used to directly query a key-pose via the identification of its defining pose features.

- **A pose rectification pipeline**: We discuss a taxonomy of typical pose estimation errors that we face when using pose estimation systems in visually challenging training videos: Joint swaps, joint detection outliers, and general estimation noise. A pipeline of optimization problems is proposed for rectifying each error class by merely arguing about joint locations. It involves the definition of a spatiotemporal graph, where vertices represent joint locations in different frames and edges connect vertices within a frame (spatial) and vertices over different frames (temporal). All edges are weighted with probabilities that reflect whether groups of joints belong to the same joint trajectories. A set of constraints defines which edges may appear together to form a trajectory and which edges are mutually exclusive. An integer linear program is used to partition the graph into joint trajectories by deleting edge with low probability from the graph. Contrary to similar approaches, we explicitly model the possibility for joints to swap their label, e.g., a left wrist detection might be reassigned to the trajectory of the right wrist after optimization and vice versa. Edge weights are determined from motion kinematics, in particular velocity and acceleration of joints. We show that the rectification of pose sequences likewise improve joint and key-pose identification.

- **A convolutional neural network for predicting the jump force of ski jumpers**: We define the problem of estimating the jump forces of an athlete as a classical machine learning question: Given that we continuously observe the pose of a person during a jump motion, can we infer the measurements of sensor plates taken at the same time? We propose a deep neural sequence-to-sequence network for learning a mapping between a sequence of poses obtained from a pose estimation network and a sequence of force measurements known at training time. Contrary to other sequence learning methods like LSTM networks, the proposed network is purely convolutional, which implicitly (via the receptive field) allows for controlling how much of an inputs signals "history" is included in the estimation of the output. Therefore, we leverage a mathematical operation called a *dilated convolution* to be able to grow a large receptive field without having to use an extensive amount of convolutional or pooling layers.

## 1.3 Publications

Parts of this thesis have been published at the following international conferences:

**Refining Joint Locations for Human Pose Tracking in Sports Videos.** *Dan Zecha, Moritz Einfalt, Rainer Lienhart.* IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops 2019. Long Beach, CA, USA, June 2019.

**A Convolutional Sequence to Sequence Model for Multimodal Dynamics Prediction in Ski Jumps.** *Dan Zecha, Christian Eggert, Moritz Einfalt, Stephan Brehm, Rainer Lienhart.* First International ACM Workshop on Multimodal Content Analysis in Sports (ACM MMSports'18), part of ACM Multimedia 2018. Seoul, Korea, October 2018.

**Kinematic Pose Rectification for Performance Analysis and Retrieval in Sports.** *Dan Zecha, Moritz Einfalt, Christian Eggert, Rainer Lienhart.* IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops 2018. Salt Lake City, USA, June 2018.

**Pose Estimation for Deriving Kinematic Parameters of Competitive Swimmers.** *Dan Zecha, Christian Eggert, Rainer Lienhart.* Computer Vision

Applications in Sports, part of IS&T Electronic Imaging 2017, Burlingame, California, January 2017.

**Key-Pose Prediction in Cyclic Human Motion.** *Dan Zecha and Rainer Lienhart.* IEEE Winter Conference on Applications of Computer Vision 2015 (WACV 2015), Waikoloa Beach, HI, January 6-9, 2015.

**Swimmer Detection and Pose Estimation for Continuous Stroke Rate Determination.** *Dan Zecha, Thomas Greif, and Rainer Lienhart.* Multimedia Content Access: Algorithms and Systems VI, part of IS&T/SPIE Electronic Imaging, 23 January 2012, Burlingame, California, USA.

## 1.4 Thesis Outline

The main part of this thesis covers five chapters. In Chapter 2, we define the task of pose estimation as the foundation of this thesis. We will especially define the term activity kinematics as a central evaluation goal in this work and also discuss the specific datasets used throughout our experimental sections. In Chapters 3 and 4, the estimation of motion parameters is investigated using different types of pose detectors and estimators. We describe a spatiotemporal pose detector for finding specific, recurring poses in Chapter 3 and use a simple probabilistic scheme to infer latent poses. In Chapter 4, we present and discuss different deep neural network architectures to estimate the full pose of an athlete and infer kinematic parameters from pose estimates directly. Chapter 5 analyses the error behavior of neural pose estimation networks. Due to the visually challenging nature of our problem domain, different classes of errors can be identified that impede the desired quality of pose estimates. A series of optimization problems is proposed, each of which tackles a different error mode. As a result, we can measure a considerable improvement in pose estimation scores. In Chapter 6, we explore the question of how the jump forces of a ski jumper can be predicted in an uncalibrated camera setting using only the output of a pose estimator using fully convolutional networks with dilated convolutional layers. The final Chapter 7 concludes the thesis.

# 2 — Foundations and Datasets

The topic of kinematic and dynamic parameter estimation in video footage cannot be discussed without a conceptional definition of the human pose in an image. In this chapter, we introduce indispensable notation and concepts concerning the human pose in images and videos as well as a definition of kinematic and dynamic parameters. We conclude this chapter with a survey of datasets and error metrics commonly used throughout this thesis.

## 2.1 Human Pose Estimation

### 2.1.1 General Terminology

The *human pose* in an image is defined by the 2d locations of major joints of the human body. Throughout this thesis, a joint is often represented by a (body) *part*. While the term *part* is used ambiguously in pose estimation literature, we adopt the most common conceptual definition of a part as the region around the location of the major joint[1]. The term *appearance* of a part commonly denotes an image patch depicting a joint at its center, while a *part detector*, *part filter* or *part model* is a parameterized model trained for detecting a joint. A part detector for a joint type should ideally return a high score at the pixel position depicting the joint and a low score in image regions where no joint is present. The number of joints considered for representing the human pose varies throughout the chapters due to the natural evolution of pose models over the years. Initially, the joint set covered the head annotated at ear height, the shoulders, elbows, wrists, hips, knees, and ankles of

[1] Although the part definition used in this work is widely adopted by the majority of pose estimation researchers, a part in the original formulation of pictorial structures for object recognition [32] was defined as the limp between two adjacent joints, therefore modeling not only size and shape of the body part, but also its orientation.

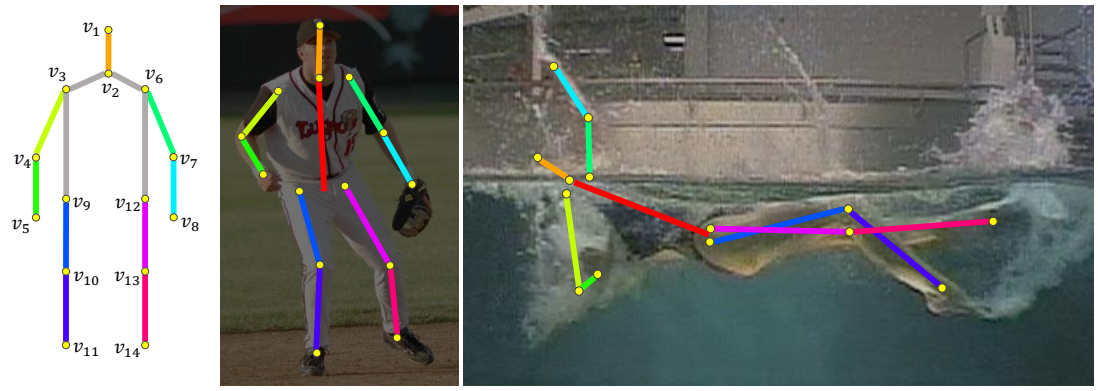



Figure 2.1: An example of a human pose graph (left) consisting of 14 joints connected according to the skeletal structure if the human body. The goal of human pose estimation is to find an instance of 2d joint locations that represent the skeletal structure of a person in an image (middle and right, middle image taken from [62]). For better clarity, the four torso connections (grey) are replaced with one visualization for the spine (red).

a person. In later publications, an annotation for the neck was added to allow for explicitly expressing the posture of the head while adapting the datasets to the state-of-the-art annotation in pose estimation research. The geometric arrangement of different parts - or rather their joint pixel positions - is denoted the *human pose*, *pose configuration*, *joint hypothesis* or *joint instance* of a person. Figure 2.1 depicts a joint hypothesis for a baseball player and a swimmer.

### 2.1.2 Human Pose Graphs

Mathematically, we represent the human pose by means of a tree-shaped graph $G = (V, E)$, where vertices $V = \{v_1, ..., v_N\}$ represent a set of $N = |V|$ body parts and edges $E \subseteq (V \times V)$ indicate which vertices are spatially related (see Figure 2.1 (left) for an example depiction). The pixel locations of the parts are defined by the joint hypothesis $L = \{\mathbf{l}_i = (x_i, y_i) : i \in \{1, ..., N\}\}$. The part $i = 1$ is usually denoted the *root part* of $G$, which is represented by a person's head location. Each vertex $v_j \in V$ relative to the root has a depth $d_j$ equal to the number of edges between itself and the root node. We will say that $v_j$ is a child of $v_i$ and $v_i$ is a parent of $v_j$ if and only if $(v_i, v_j) \in E$ and $d_j > d_i$. The children $\mathcal{C}_i \subset V$ of vertex $v_i$ are the vertices connected to $v_i$ with a depth of $d_i + 1$. Every vertex $v_i$ other than the root has exactly one parent $\mathcal{P}_i = \{v_{pa(i)}\}$ with depth $d_i - 1$, where $j = pa(i)$ is a

function returning the vertex $v_j$ that is the root of $v_i$. We collect the neighborhood of vertex $v_i$ in set $\mathcal{N}_i = \mathcal{P}_i \cup \mathcal{C}_i$.

### 2.1.3 Human Pose Estimation in Monocular Images

One of the first practical approaches to human pose estimation was proposed by Fischler and Elschlager [37], who introduced the notion of a pictorial structure representation of an object. Their model disassembles an object into a set of visual parts and spatial relationships between pairs of parts, which they describe as "spring-like" or deformable. They also present a dynamic programming scheme that computes an optimal configuration of all parts that best explains image evidence. In their seminal work, Felzenszwalb and Huttenlocher [32] expand Fischler's pictorial structures and propose a general statistical framework for object detection and human pose estimation which they denote deformable part models (DPMs). Their DMP paradigm indisputably laid one of the most influential cornerstones for object detection and articulated pose estimation research in the last decade [5, 12, 15, 42, 81, 82, 100, 109, 110]. We briefly introduce the basic notion behind deformable part models in the following.

The human pose is modeled by a tree-structured human pose graph as described in Section 2.1.2. The vertices in this graph are represented by a collection of parts. Each part encodes the visual properties of the region around a part, and pairs of parts are connected via edges. Similar to [37], the edges in this model are deformable, thus a small displacement between parts is allowed up to a certain degree, constricted by deformation terms. Felzenswalb measures how well a particular pose hypothesis fits an image $I$ using the cost function

$$\text{score}(\mathbf{l}_1, ..., \mathbf{l}_N; I) = \overbrace{\sum_{i=1}^{N} \phi_i(\mathbf{l}_i, I)}^{\text{appearance}} - \overbrace{\sum_{i,j \in E} \phi_{ij}(\mathbf{l}_i, \mathbf{l}_j)}^{\text{structure}} . \tag{2.1}$$

In this formulation, $\phi_i$ is an appearance model for the part $i$ that measures the degree of match for part $i$ at location $\mathbf{l}_i$ in image $I$. The function $\phi_{ij}(\mathbf{l}_i, \mathbf{l}_j)$ represents a deformation model that measures the degree of spatial deformation and therefore penalizes larger deviations from an optimal placement of two parts. Maximizing the cost function returns a pose hypothesis such that the appearance model $\phi_i$ yields

large scores while the deformation model $\phi_{ij}$ should encourage a joint placement with low deformation penalty.

The DPM paradigm is generic in the sense that it is independent of a specific model for the part appearance and the structure of connections between the parts. Over the years, several iconic formulations have been proposed for both components of the cost function. The appearance of a part was initially modeled by rigid template filters, which resulted from training a linear, structured support vector machine on a HoG feature [20] representation of image patches. Training patches are cropped tightly around a part, either for limbs [32, 81] or joints [5, 110]. Bourdev et al. [12] introduced the notion of a poselet as a rigid template filter for groups of joints, inspiring specific models for arms and legs [11, 42, 116]. In recent years, the explicit engineering of features for appearance modeling has been replaced by learned features based on convolutional neural networks [97, 100, 109], yielding superior detection performance over hand-crafted features by far.

The modeling of the structure has also significantly changed over the last few years. In Felzenzwalb's original formulation [32], the deformation costs

$$\phi_{ij}(\mathbf{l}_i, \mathbf{l}_j) = (\mathbf{l}_i - \mathbf{l}_j)^T M_{ij}^{-1} (\mathbf{l}_i - \mathbf{l}_j) \tag{2.2}$$

equal the Mahalanobis distance between two part placements with the restriction that the covariance $M_{ij}$ has to be a diagonal matrix. Efficiently optimizing over all possible part placements in a given image is a non-trivial problem for which Felzenzwalb formulates an efficient algorithm for a *generalized distance transform* to speed up inference. This formulation was widely adopted by most major publications in the field of object detection and human pose estimation [5, 11, 12, 42, 81, 110] at the time. With the introduction of deep convolutional neural networks (DCNNs), the paradigm for deformation scoring shifted away from explicit deformation models towards learning the allowed placement of joints directly from ground-truth pose configurations [51, 65, 95, 106].

## 2.2 Kinematics and Dynamics in Videos

We complete the discussion on types of kinematic parameters from the introduction by defining kinematic quantities that are considered in the following chapters.

### 2.2.1 Traditional Motion Kinematics

Within this thesis, we will most commonly argue about the position of joints of the human body. In a temporal context, the location $\boldsymbol{l}_i[t]$ of a joint $i$ is a vector-valued function of time

$$\boldsymbol{l}_i[t] = \begin{bmatrix} x_i[t] \\ y_i[t] \end{bmatrix} \tag{2.3}$$

where $x_i[t]$ and $y_i[t]$ are coordinate functions of time. We will use $\mathbf{l}_i^{(t)}$ to describe the Euclidean coordinate of joint $i$ in a specific frame at time $t$ and $\mathbf{l}_i$ for a location of joint $i$ in no specific temporal context. A joint in motion changes its position over time, moving along a *trajectory*. In a video, we approximate the *velocity* $\boldsymbol{v}_i^{(t)}$ of a joint $i$ in frame $t$ by

$$\boldsymbol{v}_i^{(t)} = \nabla \boldsymbol{l}_i^{(t)} = \begin{bmatrix} \frac{d}{d\tau} x_i[\tau]\big|_{\tau=t} \\ \frac{d}{d\tau} y_i[\tau]\big|_{\tau=t} \end{bmatrix} \approx \begin{bmatrix} x_i^{(t)} - x_i^{(t-1)} \\ y_i^{(t)} - y_i^{(t-1)} \end{bmatrix} \tag{2.4}$$

The velocity is a vector quantity and therefore has a direction and a magnitude of displacement. The second derivative of location is commonly known as acceleration. It is approximated by

$$\boldsymbol{a}_i^{(t)} = \nabla^2 \boldsymbol{l}_i^{(t)} \approx \begin{bmatrix} x_i^{(t)} - 2x_i^{(t-1)} + x_i^{(t-2)} \\ y_i^{(t)} - 2y_i^{(t-1)} + y_i^{(t-2)} \end{bmatrix} \tag{2.5}$$

Note that we presume that our approximations for velocity and acceleration are strictly causal, i.e., Newton's difference quotient[2] is used to approximate the gradient. Therefore, the gradient computation at time $t$ considers joint locations from frames $t-1$ and $t-2$, which entails a "lag" of $1/2$ a frame and one frame for velocity and acceleration, respectively. This lag is reflected in a larger approximation error, which can be reduced using a symmetric difference quotient[3] for approximating the derivation.

[2]Given a function $f(x)$, Newton's difference quotient is defined as $(f(x) - f(x-h))/h$ for some small $h$. The last part of Equation 2.4 uses this approximation with the smallest possible $h = 1$ for joint locations in consecutive frames.

[3]The symmetric difference quotient is defined as $(f(x+h) - f(x-h))/2h$ for some small $h$.

### 2.2.2 Activity Kinematics

We define the term *activity kinematics* as a collective term for kinematic parameters derived from activity intervals of athletes as follows. An activity interval is a motion sequence that starts when the athlete strikes a well-defined pose and ends with an equally well-defined posture. Both the start- and end-pose framing the activity interval are dubbed *key-poses*. The frame depicting a key-pose is denoted a *key-frame*.

The concept of activity intervals can be illustrated by the simple example of doing a push-ups motion. A push-up can be divided into two distinct intervals. In the first phase, an athlete starts in a key-pose where his arms are straight, followed by steadily lowering the body to the ground by bending the elbows. At the point where he reaches the key-pose with the smallest angle between both the upper arm and forearm, the first activity interval ends. In the second phase, the athlete performs the actual pushing motion, lifting his body from the ground until the arms are fully pushed through. Both phases or activity intervals together describe the complete push-up activity.

The same concept of partitioning a motion into intervals is done for world-class athletes in swimming. A cycle can be divided into partitions, where the beginning and the end of an interval is defined by a key-pose. Timing each motion interval in a cycle allows a coach to determine how well a stroke was executed and assists him to propose posture adjustments to the athlete for improving the overall performance.

A key-poses is based on a feature of the human pose or *pose feature*. Different classes of pose features can be defined based on different numbers of joints. Classical pose features include:

- The position of one joint in the image. For swimmers in a swimming channel, experts are interested in finding key-poses where the hand of a swimmer touches the water surface or where it just leaves the water. Another example is competitive swimming in a large pool, where the position of the head is used to precisely determine interim times of the swimmer crossing certain marks in a race, e.g., the 10 and 25-meter mark. In other sports like track and field athletics, this translates to identifying the locations and timestamps in a video where an athlete's foot touches or leaves the ground.

- The angle of a body part as measured in the image coordinate system. This feature involves two joints of the body. In swimming, training scientists are interested in the upper arm angles. In ski jumping, the angle of the upper and lower body, as well as the angle of skis, is the most valuable parameter for judging the jump trajectory of an athlete. This feature can also be used for the qualitative assessment of motion, for instance in high diving.

- The angle between limbs. Measuring the angle between limbs of the body commonly involves three joints. In swimming, especially for butterfly and breaststroke swimming, training scientists are interested in the angles between the thigh and lower leg as well as upper arm and forearm. Important timestamps depict occurrences of minimal or maximal angles between limbs. The same measure is of interest for ski jumpers, where the angle between upper and lower body sheds light on the goodness of posture during the flight phase.

Automating the time-consuming process of manually searching for key-poses in large video databases can be achieved using human pose estimation. In this thesis, we explore and focus specifically on the evaluation of *activity kinematics* of swimmers, which involves the precise identification of key-poses by their pose features. Therefore, experimental sections mimic real-world evaluation protocols designed by training scientists and coaches.

### 2.2.3 Dynamics

Predicting jump forces directly in videos is, of course, a difficult problem, as a force itself can only be observed if it results in an observable action. In ski jumping, this action can apparently be observed through the jump motion of the athlete. In this thesis, we formulate the estimation of jump dynamics from the motion of a ski jumper as a classical learning problem. Our task is to predict a time series representing the jump force of a ski jumper during the duration of a jump given a sequence of continuously estimated athlete poses for the same time window. Formally, we predict an output sequence of force values

$$f = (f^{(t)})_{t \in \mathcal{T}} = (f^{(1)}, ..., f^{(T)}) \tag{2.6}$$

where $(.)_{t \in \mathcal{T}}$ describes a sequence of values of length $|\mathcal{T}|$ with $\mathcal{T} = \{1, ..., T\}$. The input sequence to our learning problem will be defined by a sequence of poses $p =$

$(p^{(t)})_{t\in\mathcal{T}}$ given some learner $\mathcal{M} : P^T \to F^T$ that produces the mapping

$$f = \mathcal{M}((p^{(t)})_{t\in\mathcal{T}}).$$

Estimating dynamics by a learning problem allows to automatically find patterns connecting the observable motion with the underlying, triggering forces without having to explicitly define a physical model.

## 2.3 Datasets

The methods developed in this thesis are mainly evaluated on two distinct datasets. Training footage of swimming athletes performing in a swimming channel is used for evaluating activity kinematics. We estimate the jump forces of ski jumper in a separate dataset. Additionally, we will leverage the Leeds Sports Pose dataset (LSP, [62]) and the MSCOCO dataset [71] for initializing models prior to fine-tuning them to the other datasets. Both datasets belong to the largest, most used and evaluated datasets in the pose estimation community.

### 2.3.1 Swimming

We attempt to estimate kinematic parameters of swimmers in a swimming channel. It allows for precisely regulating the water current while observing the full posture of athletes above and below the water surface through a glass wall.

#### Terminology

The smallest unit within the repetitive motion of swimming is one *cycle* or *stroke*, defined by the time that passes between the appearance of a specific body pose and its earliest reappearance. Competitive swimming covers *four different swimming styles*, namely butterfly (fly), backstroke (back), breaststroke (breast) and freestyle (free). Figure 2.2 depicts example images for all four styles in two different swimming channels. While butterfly and breaststroke are called *symmetrical* because both halves of the body have the same posture at every time during a stroke, we denote freestyle and backstroke as *anti-symmetrical* swimming styles. The essential difference is that the arm movement in these two styles alternates from side to side; while one arm is pulling, the other one is recovering above the water line. Hence,

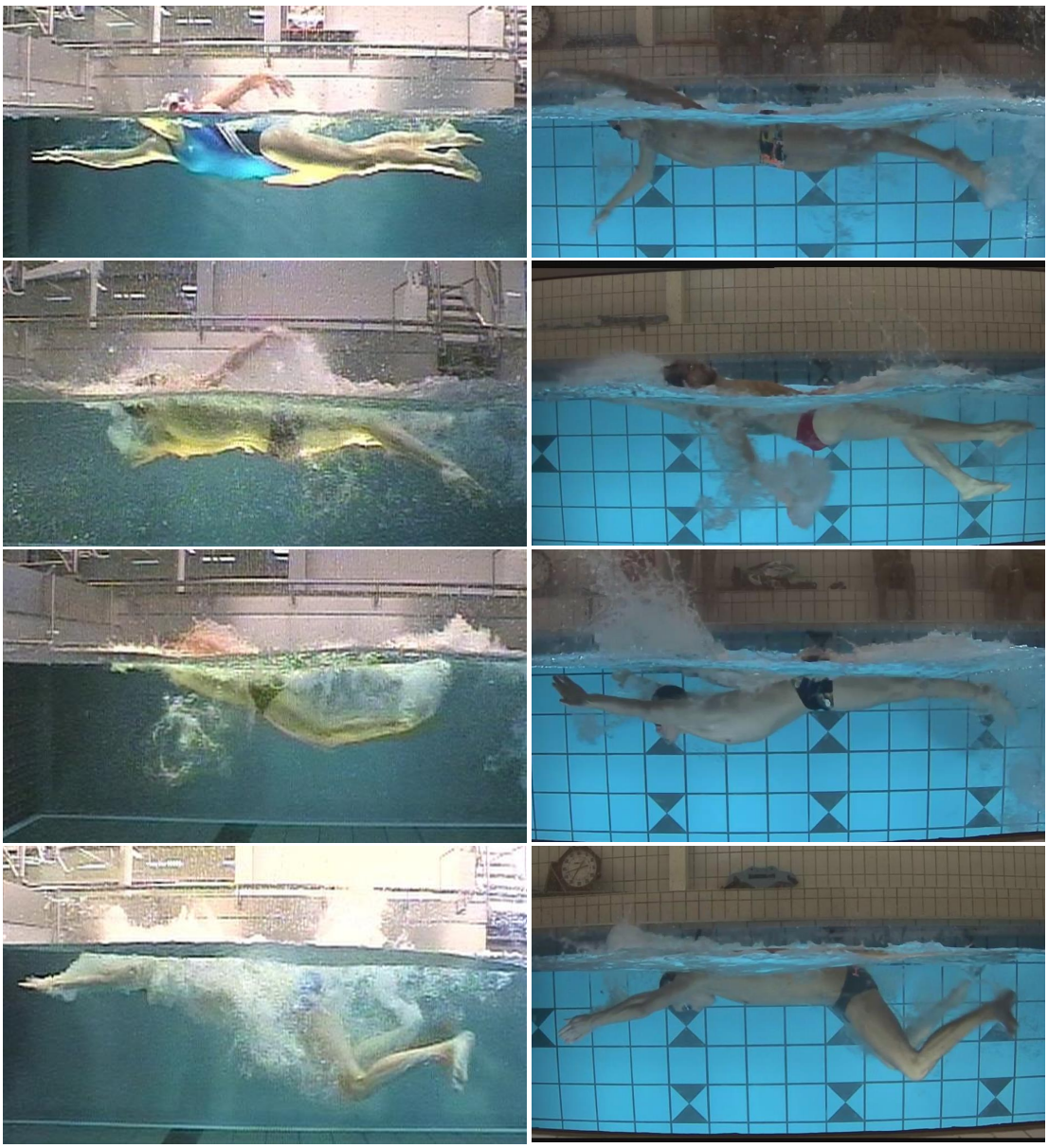


Figure 2.2: Examples images from the swimming channel dataset. The footage covers two different swimming channels located at the Institute of Applied Training Science in Leipzig (left) and the Olympic Training Center Hamburg/Schleswig-Holstein (right). The images illustrate major challenges of the dataset, which exacerbate any automated evaluation: noise from water bubbles and splashes, occlusion from refraction at the water surface, self-occlusion, foreshortening, and irregular illumination. Note for example the head in the top left image: refraction at the water line visually divides the top from the bottom, making the precise detection (and annotation) very difficult.

| | free | back | fly | breast |
|---|---|---|---|---|
| # images | 1883 | 2665 | 2100 | 2281 |
| # sequences | 24 | 32 | 26 | 28 |
| # videos | 97 | 74 | 80 | 80 |
| # key-pose annotations | 1504 | 1172 | 1920 | 1280 |

Table 2.1: A summarization of the *mmc-swim-channel* dataset. The dataset is comprised of overall 8929 images covering all four major swim styles. Each image depicts one athlete for whom we annotated the full pose. Images are organized in sequences of 50 to 100 consecutive frames. The video dataset covers overall 331 videos for key-pose estimation. In each video, the key-poses of two to four consecutive swim strokes have been labeled, usually at the beginning and again at the end of the video.

every pose of the left body half occurs approximately half a cycle later on the other side of the body. If viewed from the side, one body half in symmetric styles is mostly indistinguishable due to self-occlusion by the body half facing the camera.

### Swimmer Dataset and Evaluation Protocols

Table 2.1 summarizes the dataset used and evaluated throughout this thesis, which is comprised of images and videos. We refer to this dataset as *mmc-swim-channel*. Both modalities were recorded in swimming channels at the Institute of Applied Training Sciences in Leipzig and at the Olympic Training Center Hamburg/Schleswig-Holstein. The cameras in Leipzig record with a resolution of $720 \times 576$ pixels at $50i$, while the cameras in Hamburg record with a resolution of $1280 \times 720$ pixels at 60 frames per second. Both images and videos depict swimmers of different age, stature, gender, and body size. The footage covers different test scenarios with different water current velocities within the range of $0.8ms^{-1}$ to $2.2ms^{-1}$. All athletes are filmed from a side view through a glass wall with their left body side facing the camera, except for backstroke swimmers, who face the camera with their right body half (see Figure 2.2 for examples).

**Image Sequences.** The images in the dataset are fully annotated with complete athlete poses. Each annotation covers 14 joints, including the head, neck, shoulders, elbows, wrists, hip joints, knees, and ankles. If a joint is occluded, a human expert interpolated the most probable joint position. Only the left body side was annotated for butterfly and breaststroke swimmers due to the symmetry in their motion. Images are organized in sequences, where each sequence covers

between 50 to 100 consecutive images. Overall, we annotated the pose of athletes in 8929 images, which form 110 sequences covering all swim styles. The set of image sequences was used to train and evaluate pose estimators throughout the chapters.

**Videos.** For the task of key-pose estimation, we use a distinct set of swimmer videos. Each video is between 20 seconds and 2 minutes long and covers on average between 20 to 30 swim strokes. Videos are not annotated with ground truth poses. Instead, an expert labeled all key-frames depicting a key-pose. Key-poses where typically annotated for two to four consecutive swim strokes, often at the beginning of a video and again at the end. In some cases, it may happen that a pose can be identified as a key-pose by two different pose features at the same time. In this case, the key-pose is also annotated twice, once for each associated key-pose. Overall, we collected 331 videos covering approximately 2500 swimming cycles for which we annotated 5876 key-poses.

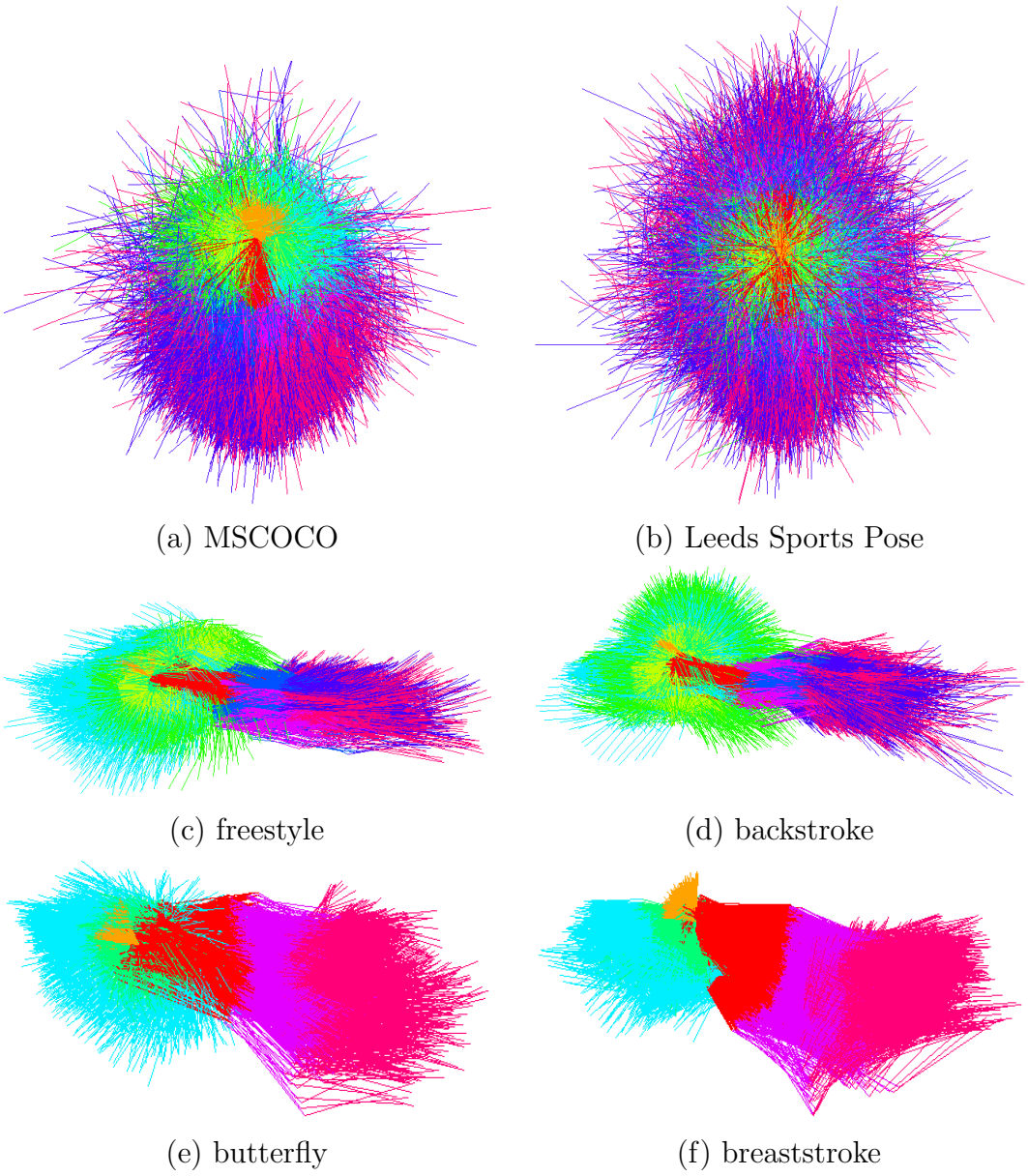
**Evaluation Protocols.** Images are used for training pose estimators in this work. *If not stated otherwise*, we always measure the performance of a pose estimator on the complete image set using 3-fold cross-validation. Therefore, the image dataset is partitioned into three almost equally sized partitions (we do *not* split sequences). Either two partitions are used for training and one part is used for testing. We split about 10% of the images from the training set and keep them for validating the pose estimator. In the case of key-pose estimation, if the proposed approach is parameterized and needs training, we use a leave-one-out cross-validation scheme, where we keep one video for testing and train the key-pose estimator on the rest of the set. The final performance is then averaged over all splits.

In Chapters 3 and 4, only a subset from the freestyle swimmer images and videos in Table 2.1 will be used for training and evaluation. We will introduce this subset as well as applicable evaluation protocols in the respective chapters.

### Peculiarities and Visualizations of Swimmer Poses

We discuss some peculiarities of swimmer poses and motion to convey a better understanding of the composition and variability of our data and to sensitize for extraordinary challenges.

**Range of Motion.** A property of sports footage that makes it more interesting to work with is that athletes tend to have a higher range of motion compared to normal people in everyday situations. Using the pose coloring scheme from Figure



(a) MSCOCO (b) Leeds Sports Pose

(c) freestyle (d) backstroke

(e) butterfly (f) breaststroke

Figure 2.3: The dynamic range of motion of different datasets. Each visualization is generated by aligning all pose configurations in a dataset at the necks, resizing all configurations such that they have approximatelly the same upper body size and then plotting them on top of each other. The coloring scheme from Figure 2.1 was used, i.e., light blue and green colors depict arms, dark blue and purple the legs, orange and red represent head and spine, respectively. The top row depicts the dynamic range of MSCOCO poses ([71], only fully annotated configurations) and the Leeds Sports Pose (LSP, [62]) datasets. The bottom two rows show all four different swimming styles in our dataset.

2.1, we visualize this circumstance in Figure 2.3. Each color representation in this depiction was generated by aligning all pose configurations in a dataset at the neck of each pose configuration, resizing all configurations such that they have approximately the same upper body size and then plotting them on top of each other. This allows us to get a better feeling for the dynamic range of motion throughout different datasets. We first compare fully annotated pose configurations from MSCOCO and LSP in Figure 2.3 (a,b). MSCOCO depicts mostly upright persons in different everyday scenarios. Compared to LSP, the range of motion in this dataset is only limited. LSP is one of the most popular datasets in articulated pose estimation research, as it covers a wide variety of flexible poses in all orientations (see Figure 3.4 for examples). Compared to these general datasets, swimmers (Figure 2.3 (c-f)) seem to express less range of motion due to their uniform general orientation. Upon closer inspection, we can see that the range of motion, especially for the arms, is equally or even more articulate and prolated.

**Swim Stroke Velocities.** To give a better understanding of the kinematic variability in swim strokes, we visualize velocity vectors of the whole left body side covering one cycle of each swim styles in Figure 2.4. This plot was generated from ground truth annotations of the left arm joints (shoulder, elbow, wrist) and the left leg joints (hip, knee, ankle). Only for backstroke swimmers, the right body side was used. We densely sampled points on the arm and leg bones in between pairs of adjacent joints. This allows us to plot the illustrated dense velocity vector fields. Each gradient is rooted at the small dots with the directional lines representing the gradient orientation. Line length and color both represent velocity magnitude. It should become clear in this visualization that motion kinematics can vary significantly depending on the the athlete, his or her pose, exhaustion level, and water current.

**Pose Estimation Challenges in Swimming.** The swimmer dataset comes with ordinary pose estimation challenges like foreshortening and self-occlusion. Additionally, new problems arise from the peculiarities of this specific setting, which are partially depicted in Figure 2.2. For example, the aquatic environment introduces considerable amounts of refraction at the water surface which regularly occludes parts of the pose to the point of complete limb occlusion during specific phases of a stroke. Refraction also visually splits body parts, where one half of a part above the water surface is clearly offset from the half below. This is illustrated by the head

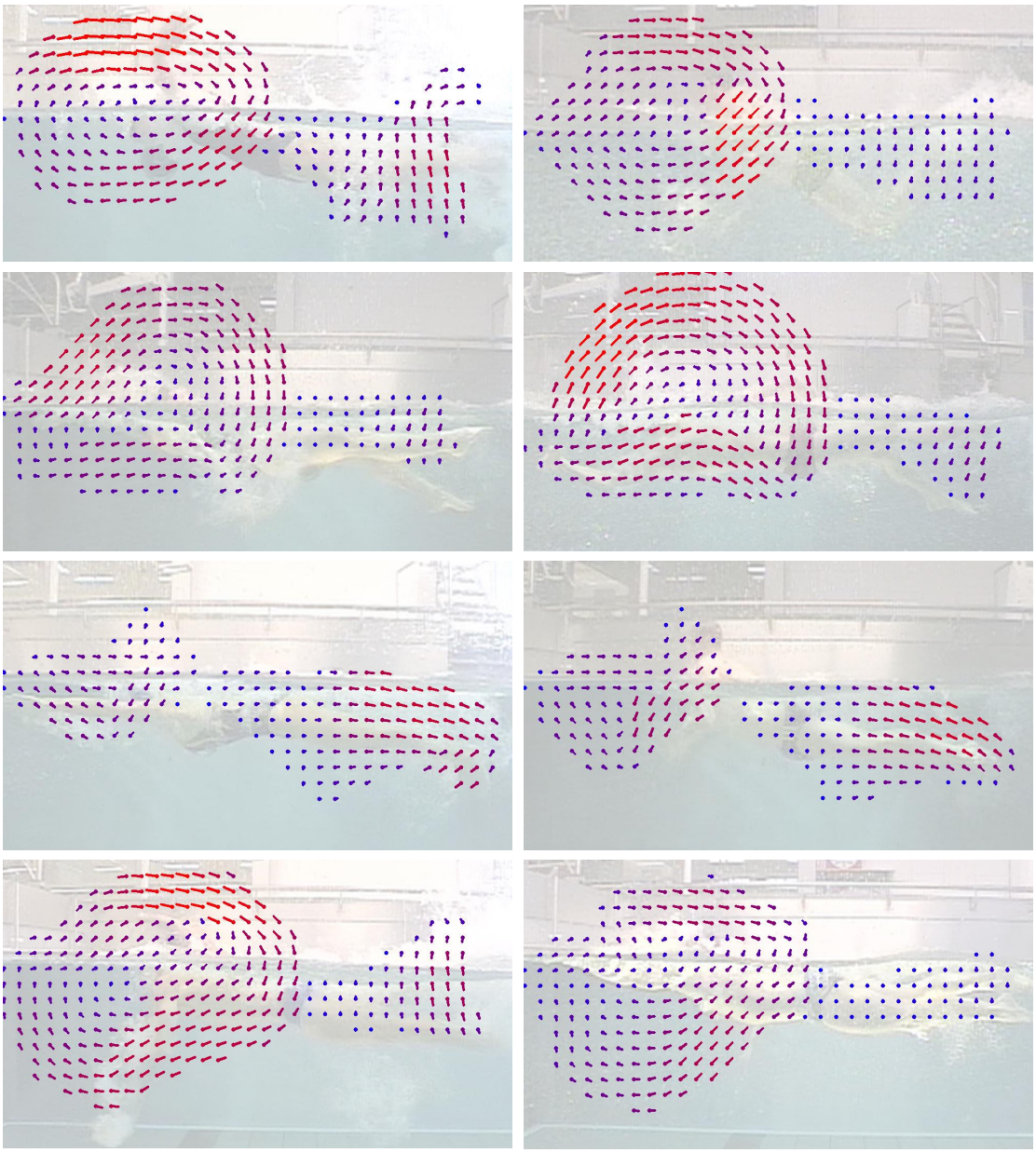


Figure 2.4: We use annotations of the arm (shoulder, elbow, wrist) and leg (hip, knee, ankle) joints of the left body side (right side for backstroke) and densely sample points in between neighboring joints. The depicted grid illustrates the velocity vectors of one complete swimming cycle as obtained by Equation 2.4. Line directions equal gradient directions, line color and length represent magnitude (red is faster). From top to bottom: butterfly, backstroke, breaststroke, freestyle. Motion kinematics vary during a swimming cycle depending on the athlete, his or her pose, exhaustion level, and the water current.

and shoulder of the swimmer in the top left image in Figure 2.2. In contrast, fast swimmers can produce waves that due to refraction allow for observing the torso from the side and the top at the same time. Adding to this, constant self-occlusion and external occlusion from water bubbles and splashes exacerbate this problem. In general, noise from water bubbles and splashes makes a precise localization of joints considerably more challenging than in non-aquatic environments. Lighting conditions can also cause problems. While lighting should be easily controllable in the confined environment of a swimming channel, we found that illumination is often less than optimal due to poor architectural design or operator mishandling during recordings.

It can be argued that our footage is visually homogeneous and that it should be much simpler to train a pose estimator that performs very well on this data compared to "in-the-wild" settings. We found that due to the aforementioned visual challenges and without the clear presence of distinct visual cues like the face of a person, pose detectors tend to have trouble correctly distinguishing between body sides. This often leads to false joint assignments, even in novel pose estimation systems. An elaborate discussion on this topic is given in Chapter 5.

### 2.3.2 Ski Jumping

For the task of predicting the jump forces of ski jumpers in videos, we put together a dataset consisting of 467 *video sets* of different athletes performing one jump in each set. One set consists of multiple camera views (examples in Figure 2.5). The different cameras are mounted along the end of the jump and along the hill and thus depict the ski jumper from either the right or left body side in different phases of a jump. The cameras are also synchronized by a light barrier that is triggered by the athlete at the beginning of the jump. The registration of the camera views was not possible with traditional methods, as the camera images do not overlap enough for classical calibration methods to be applicable. The videos were recorded in a $720 \times 576$ pixel format at 25 interlaced frames, leaving us with 50 full frames after deinterlacing. The whole footage covers a wide range of recording scenarios (day vs. night, good vs. bad vs. foggy weather, snowy and rainy scenes, winter and summer footage) and a variety of ski jumpers of all ages and statures.

Throughout all 467 videos sets, a number of 9323 athlete poses have been annotated by experts. Similar to butterfly and breaststroke swimming, ski jumpers


Figure 2.5: Examples images from the ski jumper dataset.

exhibit a symmetrical pose. Hence, only the head at ear level, the shoulder, elbow, wrist, hip, knee, and heel of the body side facing the camera were annotated. From the 9323 annotated poses, we keep 500 for testing and another 900 for the validation of the pose estimator.

Jump forces are estimated on a subset of 225 video sets for which we were able to obtain a *series of force measurements* for each video set. We split the dataset into 180 video sets for training, 20 for validation and 25 for testing. The measured force values were recorded at 2 kHz with multiple force plates which are synchronized among each other but not with the video system. The exact process of how to continuously measure the jump forces of a moving person over a distance of the last 17 meters of the ski jump is unknown to us. In the end, the measuring system outputs a low-pass filtered force series where each reading is labeled with the distance between the ski jumper and the edge of the ski jump. Force values lie in the range of 0 kilonewton[4] to 2.5 kilonewton.

[4]A kilonewton (symbol: kN) is a unit of force: $1kN = 102kg \cdot 9.81ms^{-2}$.

## 2.4 Related Work

In each chapter, we will cover the relevant scientific work related to the discussed approaches and algorithms. Enveloping the whole thesis is the topic of computer vision in sports, specifically computer vision algorithms for analyzing swimmers. In general, publications focusing on water sports and people in aquatic scenarios are still rare. Most work researching the tracking of people in aquatic environments has focused on drowning detection [27], localization of athletes in swimming competitions [89, 101], the automatic analysis of large databases of swimming videos [90] and motion analysis for video-based swimming style recognition [98].

**Pose estimation and initialization.** A Kalman filter framework is presented in [46] to explicitly model the kinematics of cyclic motions of humans in order to estimate the joint trajectories of backstroke swimmers. Ries et al [84] use Gaussian features for detecting a specific pose of a swimmer in a pool with the intention of initializing his/her pose. Einfalt et al. [26] investigate CPM architectures for improving pose estimates of swimmers in a swimming channel by conditioning the networks with the swimming style as an additional network input.

**Kinematic parameters.** Most research in kinematic parameter extraction of swimmers focuses on detecting the stroke rate. This has been explored by Victor et al. [104], who use a DCNN to predict a sinusoidal stroke signal of competitive swimmers in a swimming pool. Hakozaki et al. [49] tackle the same problem using a modified LSTM network [54]. Lienhart et al. [70] use classical data mining techniques to infer kinematic parameters like the stroke rate from matching pose configurations in a large dataset of automatically estimates poses. Fani et al. [30] perform a qualitative analysis of the pull motion of freestyle swimmers by evaluating human pose estimates.

**Key-poses.** The concept of key-poses has also been investigated in different sports scenarios. Given perfectly detected poses, Dios et al [23] automatically determine key-poses in cyclic human motion by performing a principal component analysis on pose estimates, followed by a supervised cluster analysis, which allows for classifying the quality of stretch motions with high precision. Likewise, Vicente et al [24] automatically pick key-poses from a query video with pose annotations obtained through motion capture in order to classify motion sequences for Taekwondo videos using a latent-dynamic conditional random field. Both approaches

presume an almost perfect annotation of the human pose. Carson et al [14] includes a selection process for action specific postures by matching shape information from individual frames in order to recognize specific tennis strokes in game footage.

Each chapter in this thesis includes its own related work section presenting papers specifically relevant to the chapter.

# 3 — Activity Kinematics from Spatiotemporal Poselet Activation

We address the question of how to estimate activity kinematics via key-pose identification using a pose detection approach in this chapter[1]. Note the difference between articulate pose estimation and pose detection: While we try to identify all joints of an athlete with a pose estimator, we attempt to classify complete poses using a pose detector, which is somewhat related to the field of object detection with the "object" being a specific pose. In visually challenging training scenarios, the detection of a key-pose can be difficult because either the associated key-pose feature is too inconspicuous or the estimation approach itself performs subpar for that pose. As a consequence, we might assume that key-poses are *latent* or "hidden", implicating that we are not able to detect them directly. We address the challenge of latent key-poses identification by leveraging the fact that cyclic motion like swimming or running has a predetermined motion structure. Additionally, we make the assumption that all motion sequences have at least some expressive poses which can be identified using a discriminative pose detector. We label these expressive poses *support poses*, as they *support* the discovery of latent key-pose occurrences.

The proposed method can be summarized as follows: We train discriminative pose detectors, which consist of an ensemble of smaller detectors called poselets. Each poselet is trained for detecting the presence of specific arm configurations of the athlete. Therefore, a poselet is trained from a *spatiotemporal* set of image patches, i.e., the training data for one poselet detector includes patches depicting the same configuration from different swimmers (spatial closeness) but also configu-

[1]The work presented in this chapter was partially published in [115, 116].

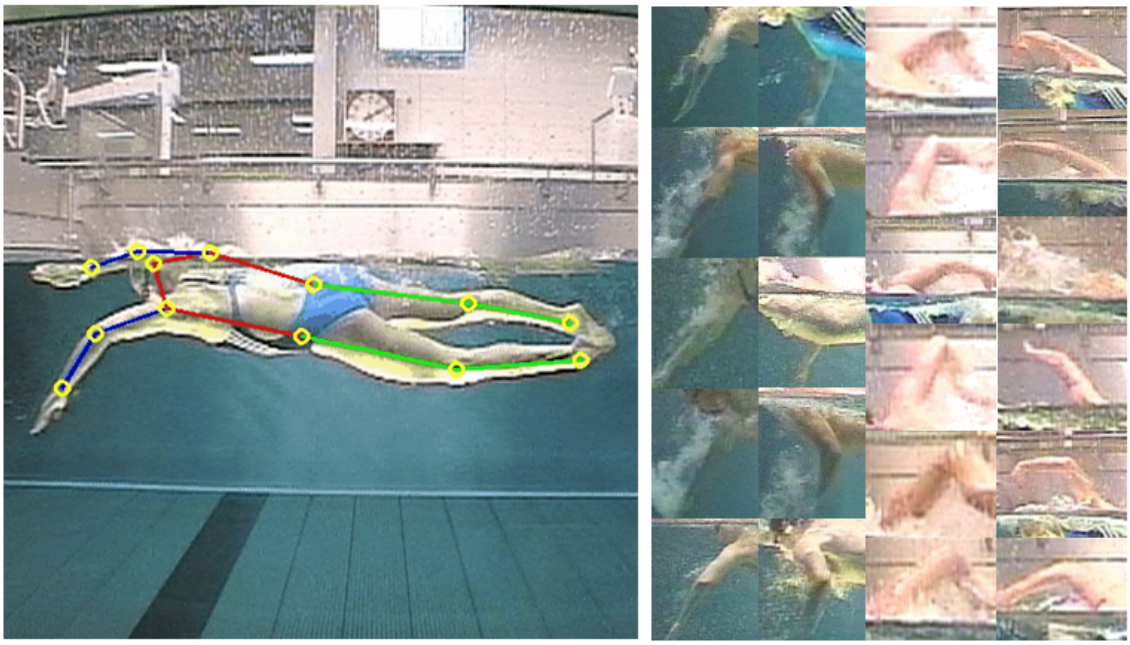


Figure 3.1: Left: A swimmer in a swimming channel. Joints are depicted by yellow circles. Arm configurations, depicted in blue, are used for poselet training. Right: A selection of training patches of arms. Each column depicts patches representative for one poselet cluster.

ration patches from the immediate surrounding frames (temporal closeness). Patch examples are depicted in Figure 3.1. The whole poselet ensemble is used to identify support poses within the cyclic motion of a swimmer. A measure of goodness can be defined for identifying the most suitable support poses, which are then used to estimate the occurrence of key-poses in a simple interpolation scheme. The proposed approach is evaluated on a subset of the *mmc-swim-channel* dataset containing only freestyle swimmers.

## 3.1 Related Work

**Pose Estimation With Part Based Models.** Part based models have played a huge role in the fields of object detection and (human) pose estimation within the last years. Based on the fundamental work Fischler and Elschlager [37], these models represent an object through multiple parts which are connected via deformation terms, allowing for matching them in a flexible configuration. Various manifes-

tations of this basic notion have been developed through the years, kicked off by Felzenszwalb et al [33] with their deformable part models for object detection. Different refinements have been proposed specifically for human pose estimation, e.g. by enriching a model with additional parts to compensate for the flexibility of the human body [110] or by allowing rotation of parts [5].

While effective implementations of part detectors have been proposed for characteristic body parts like head and torso, part templates for extremities are usually weak. This issue has been addressed by [81], who argue that person and body part templates should be pose specific rather than generally trained. Bourdev et al [12] also follow this notion by proposing the concept of poselets, generic rigid part detectors based on Histograms of Oriented Gradients (HoG) [21] as a generalization of specific body part detectors. Poselets lift the spatial limitation of parts being connected to an actual body part and encode generic parts of the body. Gkioxari et al [42] recently utilized poselets for training discriminative classifiers to specifically differentiate between arm configurations of a person.

**Poselets in Videos.** In the context of key-frame selection in videos, poselets have been used for human activity recognition. Raptis et al. [83] propose a framework based on poselet activations for selecting key-frames that represent key-states in an action sequence. An additional classifier trained on pairwise terms for all activations then decides if a specific action sequence occurred. Carson et al. [14] select action specific postures by matching shape information from individual frames in order to recognize specific tennis strokes in-game footage.

**Human Gait Analysis.** The analysis of human gait is the most established application in the field of periodic motion research. A big focus lies on the identification of a person via his/her intrinsic gait signature, for example by determining and tracking the body shape [118] or through the fusion of multiple gait cycles [55]. More general approaches strive to recover the human body pose [48] in order to retrieve a full set of gait parameters. Periodic motion in images was examined by Cutler and Davis [19], who use self-similarity and frequency transformations to obtain the frequency of the periodic motion of human and animal gait.

**Procrustes Analysis.** The problem of matching configurations and shapes was extensively studied in the past under the name orthogonal Procrustes Statistics [44, 87, 92]. Procrustes Analysis has applications in rigid body alignment, crystallography, and multivariate analysis methods, in particular, Multidimensional scaling

[67, 99]. The derivation of the semimetric for spatiotemporal configuration matching in this chapter particularly builds on the work of [88].

## 3.2 Spatiotemporal Poselet Detection

The deduction of key-poses from support poses in a predictable or repetitive motion sequence is a two-stage process. Firstly, an ensemble of poselets is trained for the identification if support poses. Secondly, a maximum likelihood estimator is proposed for good support poses in order to predict the occurrence of a key-pose.

### 3.2.1 Poselet Training

We build our system on localizable parts of the human pose, initially introduced as poselets by Bourdev et al. [12]. The original work defines poselets as rigid linear filters, based on Histograms of Oriented Gradients (HoG, [21]) features. Each filter is trained from a set of example image patches that are close in configuration space. The patches are transformed into feature space and a linear Support Vector Machine (SVM) is trained to classify patches and thereby learn a representation. For inference, the SVM is reformulated as a dense linear filter, which is cross-correlated with the HoG feature representation of a test image, yielding a classification score for every placement of the filter.

The proposed approach depends heavily on precisely trained, distinguishable poselets. We achieve this by extracting patches of characteristic parts of the poses, e.g., patches of the limbs from all images. The underlying ground truth joint annotations are used to cluster the image patches into groups depicting similar configurations (see Figure 3.1 for examples). From each group, one poselet classifier is trained.

A simplifying assumption often made in the context of poselets is that the visual representation of a part does not change with part rotation. Hence, other approaches like [42] strive to find the best possible transformation between different configurations by rotating, reflecting, translating, and rescaling body part configurations. In contrast, we extend the purely spatial representation of a poselet and additionally include configurations from a small time window around a pose configuration. We develop a distance semimetric which explicitly excludes rotation and

reflection of arm and leg configurations, therefore preserving the temporal context of configuration patches.

**Dataset**

We use a subset of the *mmc-swim-channel* dataset covering 1200 images of freestyle swimmers with fully annotated poses for training the poselets. All images come from footage exclusively recorded in the Leipzig swimming channel. The images cover different athletes, three male and five female, performing overall 20 strokes. From these images, groups of "sub-configurations" are extracted, e.g. arm-configurations (shoulder, elbow, and wrist of the same arm) or leg configurations (hips, knees, and angles). These sub-configurations are clustered using k-means and the following distance measure.

### 3.2.2 Temporal Poselet Clustering

For the purpose of this section, let $A' = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n)^T \in \mathbb{R}^{n,2}$ be a configuration of $n$ joints, where each $\boldsymbol{a}_i \in \mathbb{R}^2$ $(i = 1, \cdots, n)$ denotes a 2-dimensional location $(x, y)$ of one joint. The ordering of rows in each configuration is used to label the points, e.g., $\boldsymbol{a}_1$ always denotes the head coordinate. Similarly, let $B'$ be a different configuration. We normalize each configuration by

$$A = \frac{A'}{\sqrt{tr[A'A'^T]}}, B = \frac{B'}{\sqrt{tr[B'B'^T]}} \tag{3.1}$$

where $tr[.]$ denotes the trace of a matrix. It follows that

$$tr[AA^T] = tr[BB^T] = 1 \tag{3.2}$$

holds for all configurations in the dataset. The importance of this normalization will become clear in the following derivation. A measure for the distance between two rigid configurations is given by Procrustes statistics [44, 88, 92],

$$\begin{aligned} d_1(A, B) &= \min_{s,R,c} \sum_{i=1}^{n} \parallel \boldsymbol{a_i} - sR^T \boldsymbol{b_i} + \boldsymbol{c} \parallel_2^2 \\ &= \min_{s,R,c} \parallel A - sBR + \mathbf{c} \parallel_2^2 \end{aligned} \tag{3.3}$$

where $s$ is a scaling factor, $R$ models an optimal rotation and reflection and $c$ denotes a translational offset between configurations $A$ and $B$. Optimizing this measure performs a transformation on $B$ to match it as best as possible with $A$, leaving the sum over residuals as a final measure-of-goodness between both configurations. A solution to this problem is given by Gower [44],

$$d_1(A,B) = tr[\bar{A}\bar{A}^T] - \frac{\{tr[(\bar{B}^T\bar{A}\bar{A}^T\bar{B})^{1/2}]\}^2}{tr[\bar{B}\bar{B}^T]} \tag{3.4}$$

where $\bar{A}, \bar{B}$ are mean centered versions of $A$ and $B$, respectively.

We formulate a modified measure by setting $R = I$ to the identity, which yields

$$d_2(A,B) = \min_{s,c} \parallel A - sB + \mathbf{c} \parallel_2^2 . \tag{3.5}$$

The configuration $B$ is translated via $\boldsymbol{c}$ and resized with a scaling factor $s$. This formulation closely resembles the classical Procrustes optimization problem with the important difference that we do not allow for $B$ to be rotated and reflected by a linear transformation. This will assure that a configuration from any point within the cyclic motion is not reflected or rotated onto a configuration that is not temporally close. It can be shown [88] that subtracting configuration means $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ from $A$ and $B$ respectively minimizes equation 3.5 w.r.t. the translation $\mathbf{c}$. Hence, we define $\bar{A} = (\boldsymbol{a}_1 - \bar{\mathbf{a}}, \cdots, \boldsymbol{a}_n - \bar{\mathbf{a}})^T$ and can set $c = 0$. As a scaling operation does not change the mean of a configuration, we can rewrite Equation 3.5 to

$$\begin{aligned} d_2(\bar{A},\bar{B}) &= \min_s \parallel \bar{A} - s\bar{B} \parallel_2^2 = \min_s tr[(\bar{A} - s\bar{B})(\bar{A} - s\bar{B})^T] \\ &= \min_s (tr[\bar{A}\bar{A}^T] + s^2 \cdot tr[\bar{B}\bar{B}^T] - 2s \cdot tr[\bar{A}\bar{B}^T]) \end{aligned} \tag{3.6}$$

We minimize this expression w.r.t. $s$ by differentiating and solving for $s$:

$$s = \frac{tr[\bar{A}\bar{B}^T]}{tr[\bar{B}\bar{B}^T]}. \tag{3.7}$$

When solving the full Procrustes problem in Equation 3.3, Sibson et al. [92] prove that $s$ is strictly non-negative. By setting $R = I$ we can no longer guarantee that $s \geq 0$ still holds. A negative scaling factor is equivalent to a point reflection of the configuration at the coordinate origin with additional scaling. As we require the optimization problem not to reflect any configuration, we can enforce a positive

scaling factor $s_p = |s|$ and substitute into Equation 3.6:

$$\begin{aligned} d_3(\bar{A}, \bar{B}) &= tr[\bar{A}\bar{A}^T] + \left|\frac{tr[\bar{A}\bar{B}^T]}{tr[\bar{B}\bar{B}^T]}\right|^2 \cdot tr[\bar{B}\bar{B}^T] - 2\left|\frac{tr[\bar{A}\bar{B}^T]}{tr[\bar{B}\bar{B}^T]}\right| \cdot tr[\bar{A}\bar{B}^T] \\ &= tr[\bar{A}\bar{A}^T] + \frac{tr[\bar{A}\bar{B}^T]^2}{tr[\bar{B}\bar{B}^T]} - 2\left|\frac{tr[\bar{A}\bar{B}^T]}{tr[\bar{B}\bar{B}^T]}\right| \cdot tr[\bar{A}\bar{B}^T] \end{aligned} \tag{3.8}$$

We may utilize the fact that configurations are normalized as constituted in Equation 3.2 and simplify to

$$d(\bar{A}, \bar{B}) \equiv d_3(\bar{A}, \bar{B}) = \begin{cases} 1 - tr[\bar{A}\bar{B}^T]^2 & \text{if } tr[\bar{A}\bar{B}^T] \geq 0 \\ 1 + 3tr[\bar{A}\bar{B}^T]^2 & \text{else} \end{cases} \tag{3.9}$$

While this formulation might seem a bit cumbersome, we will demonstrate that it has some interesting properties.

**Properties**

An intuitive insight into properties of $d(\bar{A}, \bar{B})$ can be given by introducing a vectorized representation of a configuration. To that, it is straightforward to show that for any configuration $C$ and $D$, $tr[CD^T] = \langle \mathbf{c}, \mathbf{d} \rangle$ with $\mathbf{c} = [\mathbf{c}_1^T, ..., \mathbf{c}_n^T]^T$, $\mathbf{d} = [\mathbf{d}_1^T, ..., \mathbf{d}_n^T]^T$, and $\langle ., . \rangle$ denotes the inner product of two vectors. Given that all configurations are normalized such that $tr[CC^T] = tr[DD^T] = 1$ holds, we know that $\mathbf{c}$ and $\mathbf{d}$ are unit vectors. By arguing over dot products of unit vectors, it is now a bit more intuitive to show important properties of $d(\bar{A}, \bar{B})$:

- Non-negativity: As $\langle \mathbf{a}, \mathbf{b} \rangle \in [-1, 1]$, it follows that $d(\bar{A}, \bar{B}) \in [0, 4] \geq 0$
- Identity: Given that $\langle \mathbf{a}, \mathbf{a} \rangle = 1$, it follows that $d(\bar{A}, \bar{A}) = 0$
- Symmetry: From $\langle \mathbf{a}, \mathbf{b} \rangle = \langle \mathbf{b}, \mathbf{a} \rangle$, it follows that $d(\bar{A}, \bar{B}) = d(\bar{B}, \bar{A})$
- Triangle inequality: Given any three configurations $\{\mathbf{a}, \mathbf{b}, \mathbf{c} | \langle \mathbf{a}, \mathbf{b} \rangle = 0 \wedge \langle \mathbf{b}, \mathbf{c} \rangle = 0 \wedge \langle \mathbf{a}, \mathbf{c} \rangle = -1\}$, we can show by evaluating $d(\bar{A}, \bar{B}) + d(\bar{B}, \bar{C}) = 1 + 1 \ngeq 4 = d(\bar{A}, \bar{C})$ that the triangle inequality does not hold.

With the triangle inequality not holding, $d(\bar{A}, \bar{B})$ is not a valid distance metric. Instead, a metric with these properties is commonly referred to as a semimetric.

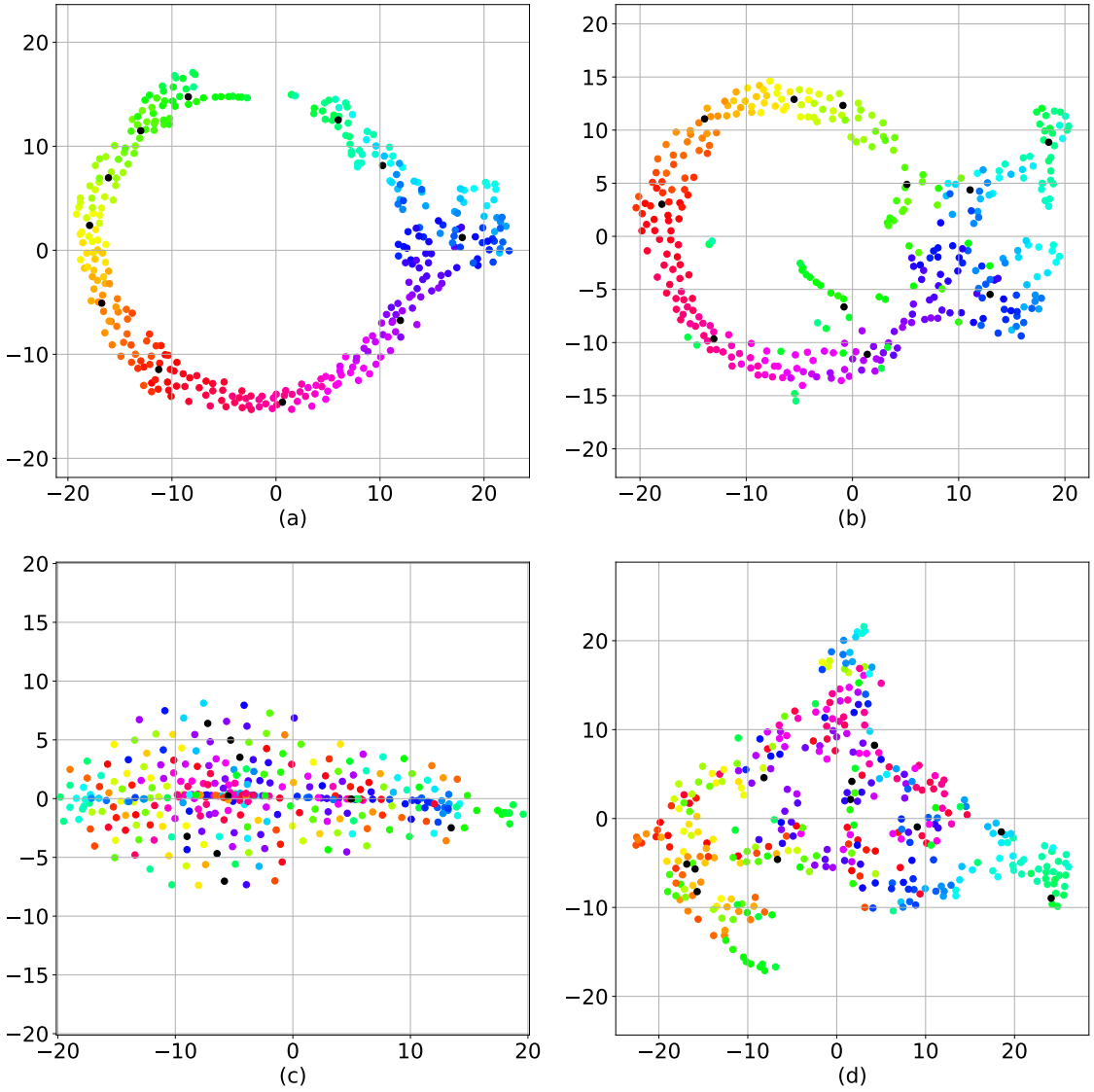



Figure 3.2: A topological visualization of different distance (semi)metric embeddings into two dimensions using t-SNE [103]. Each dot represents a configuration. If two configurations are close in configuration space according to a distance metric, they will be also close in these embeddings. Dots with similar color are temporally close within a swimming cycle, black dots indicate cluster centers. (a) The optimal solution $d_3(\bar{A}, \bar{B})$ preserves temporal structure well. (b) Minimizing $d_2(\bar{A}, \bar{B})$ retains some temporal closeness, but negative scaling factors introduce reflected configurations. (c) The full optimization of $d_1(\bar{A}, \bar{B})$ preserves no temporal structure. (d) Another distance metric from Lienhart et al. [70] preserves groups of local configurations.

### Semimetric Visualization

Figure 3.2 depicts the distances between samples from a set of 360 left arm configurations of different freestyle swimmers using different metrics. For the purpose of visual evaluation, we use t-distributed Stochastic Neighbor Embedding (t-SNE, [103]) to project all six-dimensional vectorized configurations (3 joints $\times$ 2 coordinates x and y per joint) to two dimensions. t-SNE minimizes the KL-divergence between the similarities of high-dimensional data - encoded with conditional probabilities between data pairs - and its projection to some low-dimensional representation, thereby preserving relative distances between features in the embedding. The scatter-plots in Figure 3.2 give us a good intuition about the distance between configurations in the original space. All configuration embeddings are colored such that configurations that are temporally close within a swimming cycle have similar colors. As becomes evident in this depiction (top-left), arm configurations that are temporally close in the training data are also grouped together by our distance measure, which was the intention of designing this function. When the scale in Equation 3.7 is not forced to be positive (top-right), we see that some configurations mix with temporally distinct ones. When using HoG-based pose classifiers, we found that mixing visually discriminative training patches from a noisy environment leads to subpar classification results. For comparison, we additionally visualize the full Procrustes distance $d_1$ and another distance measure defined by Lienhart et al. [70] for matching pose configurations.

### Poselet Similarity

We present some qualitative examples of similar swimmer poses using semimetric $d(\bar{A}, \bar{B})$ in Figure 3.3. Each row depicts colored arm configurations that are close in the metric space. The first image per row was used as a query image, the following images are the closest ones from the dataset under $d(\bar{A}, \bar{B})$.

The question may arise how $d(\bar{A}, \bar{B})$ can be used in more general data sets. While we deduced this semimetric with the intention to cluster temporally close configurations in cyclic (swimming-) motion, it can also be used to find poselets with visual similarity in different scales. We show this in Figure 3.4 for the LSP data set. The composition is the same as in 3.3. It becomes apparent that the proposed semimetric is also an appropriate tool for pose mining tasks, where we wish to find configurations that are most similar to a query pose.

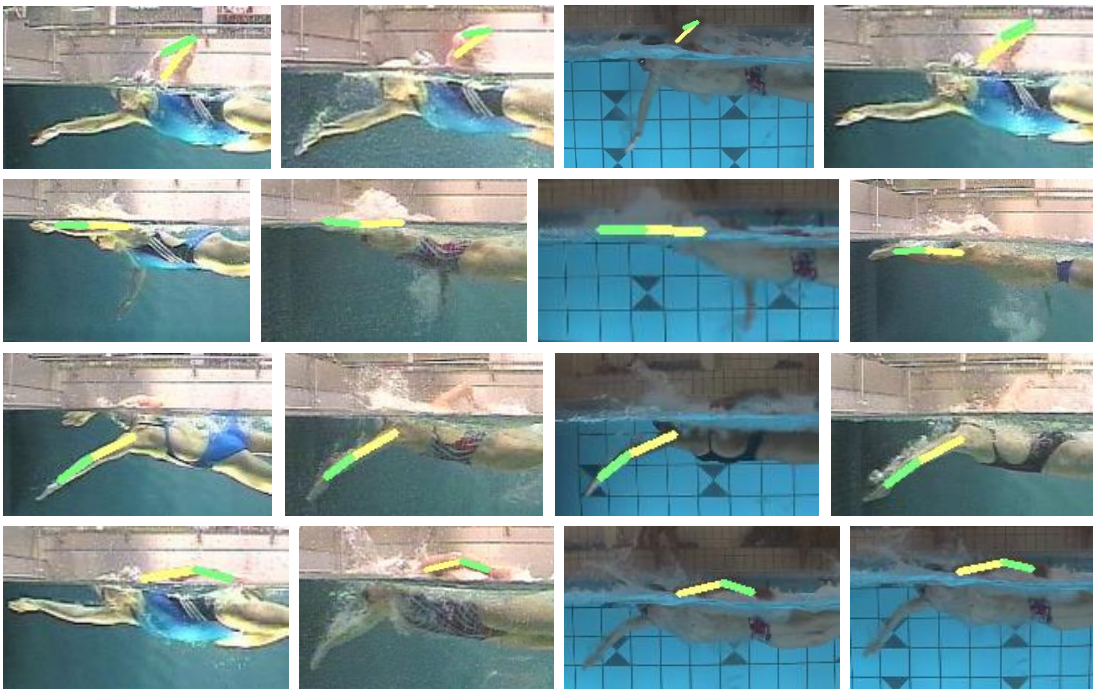


Figure 3.3: Poselet similarity examples from the swimmer data set using semimetric $d(\bar{A}, \bar{B})$. Each row depicts images with similar limb configurations. The first image in each row was used as a query image (queried configuration in colored lines), the following images are the closest one under $d(\bar{A}, \bar{B})$.

.

### 3.2.3 Classifier Training

Given a cluster of partial configurations, a patch in image $I^{(t)}$ with framenumber $t$ is defined by a location tuple $l_p = (x, y, s, w, h)$, where $(x, y)$ denotes the pixel position, $s$ defines a scale on which the patch is extracted, and $w \times h$ describe width and height, respectively. Training image patches $I^{(t)}(l_p)$ are cropped and resized to a reference poselet size. We use Histograms of Oriented Gradients [20] for feature representation $\Phi$ and transformed all patches into a feature space $\Phi^{(t)}(l_p) \equiv \Phi(I^{(t)}(l_p))$. The set of positive training patches depicting configurations is completed with a set of negative training patches of the same size, cropped from arbitrary background images. The whole dataset is used to train a linear support vector machine for classification similar to [34]. The resulting linear classifier $\beta_i$ with $i = \{1, ..., n\}$ can be reshaped to the same dimensions as $\Phi$ and is subsequently denoted a **poselet** or **poselet classifier**. It is used for patch classification by performing a cross-correlation between poselet and an image transformed to feature

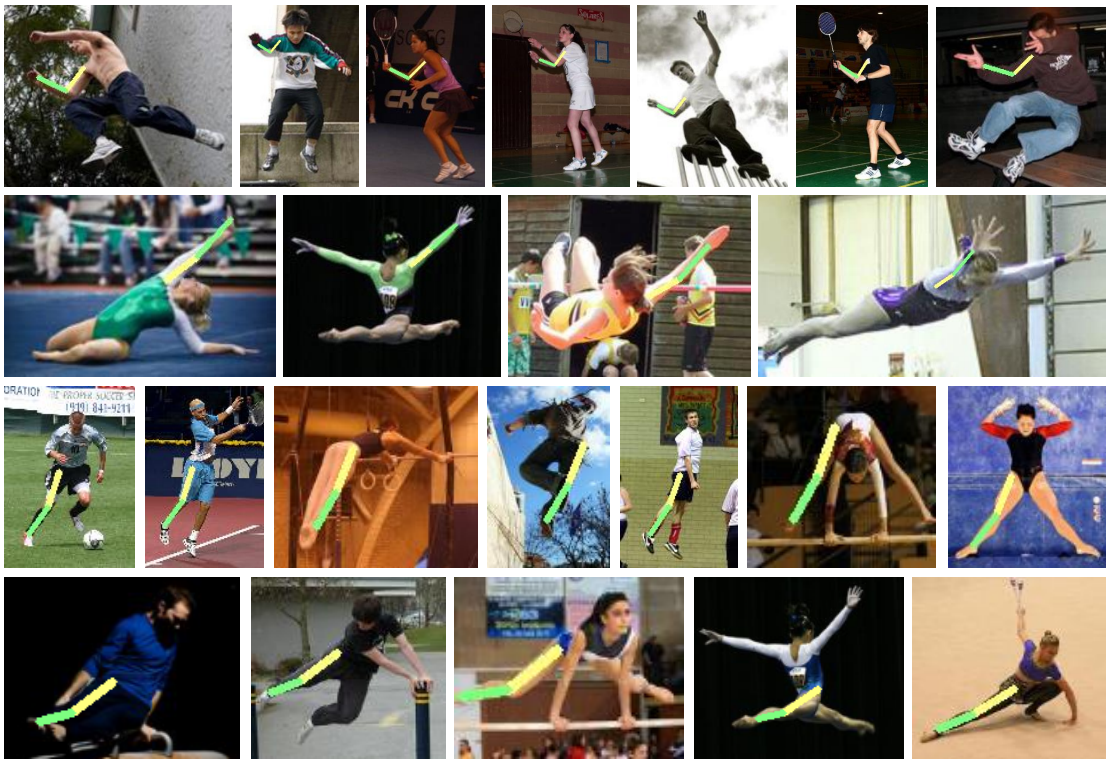


Figure 3.4: Poselet similarity in the LSP data set. Each row depicts images with similar limb configurations. The first image in each row was used as a query image (queried configuration in colored lines), the following images are the closest ones under $d(\bar{A}, \bar{B})$. Note how poselet configurations are often typical for one activity (e.g., playing tennis, gymnastics, jumping).

space $\Phi$. A poselet classifiers build from HoG features often dubbed a (linear) HoG-filter. Note however that the key-pose identification method described in the following section is not limited to this feature representation. Advanced machine learning algorithms like DCNNs can be used for patch classification as long as they return some confidence values.

### 3.2.4 Spatiotemporal Poselet Mixtures

Let $\boldsymbol{\beta}_i$ describe one of $n$ poselet classifiers of size $h_i \times w_i$. Multiple poselets are combined into a mixture $M = (\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n)$. We define a scoring function $\Gamma$ of a poselet at location $l$ in a feature pyramid[2] of frame $t$ via the cross-correlation

[2] A feature pyramid is comprised of resized versions of an input image, where each version is transformed into the feature domain $\Phi$. This technique allows for detecting objects of different sizes in images.

operator $\star$,

$$\Gamma(\boldsymbol{\beta}_i, \Phi^{(t)}(l)) = \boldsymbol{\beta}_i \star \Phi^{(t)}(l). \tag{3.10}$$

The position $l_p = (x, y, s, w, h)$ defines two coordinates $x$ and $y$ for a poselet placement, a scale $s$ of a pyramid level and the size of the sub-window $w \times h$, which equals the size of the poselet.

Similar to [21], a poselet $\boldsymbol{\beta}_0$ for the whole configuration of an athlete is trained and used to retrieve an initial hypothesis $l^*$ for the placement of the athlete by maximizing

$$l_0^* = \arg\max_l \Gamma(\boldsymbol{\beta}_0, \Phi^{(t)}(l)). \tag{3.11}$$

In our scenario, only one person is depicted per frame. In a multi-person scenario, $\boldsymbol{\beta}_0$ can be used to track multiple athletes and derive one poselet activation vector per frame and detection. If $\Gamma(\boldsymbol{\beta}_0, \Phi_t(l_0^*)) > \tau$ for a model specific detection threshold $\tau$, a good initial position of the athlete was found. This best scoring detection $l_0^*$ is projected to the original image size by dividing by $s^*$, thus

$$l_0 = (x^*/s^*, y^*/s^*, s^*/s^*, w^*/s^*, h^*/s^*) = (x_0, y_0, 1, w_0, h_0) \tag{3.12}$$

Given this root hypothesis of the location of the athlete, a score $\Gamma(\boldsymbol{\beta}_i, \Phi^{(t)}(l_i^*))$ for each poselet $\boldsymbol{\beta}_i$ in the mixture is derived by maximizing over

$$l_i^* = \arg\max_{l_k \in R} \Gamma\left(\boldsymbol{\beta}_i, \Phi^{(t)}(l_k)\right), \tag{3.13}$$

where

$$R = \left\{ l_k \middle| \boldsymbol{z}^T \left(s_k^2 \Sigma_i\right)^{-1} \boldsymbol{z} < \gamma^2 \right\} \tag{3.14}$$

and

$$\boldsymbol{z} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - s_k \left( \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \boldsymbol{\mu_i} \right). \tag{3.15}$$

$R$ is called the *activation region* and restricts the evaluable location of a poselet by means of a $\gamma$-thresholded, squared Mahalanobis distance. All positions in $R$ lie within an elliptic region shaped by deformation covariance $\Sigma_i$, which is centered at deformation anchor $\boldsymbol{\mu_i}$ relative to the root location $l_0$. The size of this region is restricted by $\gamma$, which we empirically set to $\gamma = 3$, i.e., 3-times the standard deviation of $\Sigma_i$, thus covering 99.7% of the training patch locations.

#### Deformation Terms

Both $\boldsymbol{\mu_i}$ and $\Sigma_i$ can be estimated directly from the training data by fitting a normal distribution on pairwise offsets between the root model and poselet training patch locations. Given a set of pairwise correspondences between the root model and a poselet, we want to take the athlete's size into account prior to the Gaussian parametrization. Therefore, we utilize a normalization coefficient $q = d_{ref}/w_0$, where $w_0$ is the width of the root detection at $p_0$ and $d_{ref} \in \mathbb{R}^+$ is a constant reference length, set to average over different swimmer bounding box widths. Each offset between root position and poselet patch position is normalized with $q$ prior estimation $\boldsymbol{\mu_i}$ and $\Sigma_i$. For inference, the swimmer's size is determined through the root hypothesis and averaged over multiple frames for stability. The deformation parameters in Equation 3.14 and 3.15 are then adapted by multiplying the $\boldsymbol{\mu_i}$ with $q^{-1}$ and $\Sigma_i$ with $q^{-2}$.

The model formulation above closely resembles a star-shaped pictorial structure of poselets, where all parts are connected via deformation features $\boldsymbol{\mu}$ and $\Sigma$ with a root model. Deformable Part Models [35] for object detection implement a structurally related approach, which was expanded by many others [5, 81, 110]. While the original approach penalizes the score of a poselet if it is detected too far from its "ideal" position relative to the root template, we instead perform a max pooling on all scores in the region $R$.

### 3.2.5 Key-pose Estimation from Poselet Activation

The poselet ensemble is built from temporally distinctive poselets, thus each poselet in an ensemble vividly acts like a sensor measuring the presence or absence of a body part, resulting in high or low scores over time. If the body part that the poselet was trained for is present in a frame at the location specified by the deformation parameters, the score of the poselet should be high. If the athlete continues his movement, the poselet score decreases. The underlying assumption here is that the poselet is sound, i.e., that the observed score reflects the actual image content. This may not be the case for all poselets. While some configurations are simply not suited to be represented and reliably detected by a poselet, other poselets reproduce certain motion velocities and the associated change in human posture, while a different set of filters might be adapted to specific athletes or body types. As a consequence, not

all poselets in a mixture work equally well for different athletes. Another peculiarity arises with a unique perspective on the swimming channel, where the swimmer is depicted from a side view. We found that dense HoG templates trained for anti-symmetrical swimming styles are not able to distinguish between left and right arm configurations of an athlete. Consequently, the time series of scores of a good poselet has two peaks within one cycle for anti-symmetrical swimming styles. This fact has to be considered during evaluation accordingly.

### Support Poses from Poselet Activation

The term *poselet activation* was initially defined by different authors [11, 83]. In the original definition, a set of poselets is correlated with a feature pyramid and the maximal scores of the poselets are combined into a poselet activation vector, where each vector element equals the score of the poselet if it is activated, or zero if it is deactivated. A poselet is activated if its classification score lies above a threshold and it is not activated otherwise [11].

We define poselet activation as follows. Let a timeseries of poselet scores $P_i$ for poselet $\boldsymbol{\beta}_i$ over a sequence of frames $\mathcal{T}$ be defined as

$$P_i = (\Gamma(\boldsymbol{\beta}_i, \Phi^{(t)}(l_i^*)))_{t \in \mathcal{T}} \tag{3.16}$$

In order to compensate for the noisy output of linear HoG filters, we smooth each $P_i$ with a Gaussian filter. The *activations of a poselet* $i$ are collected in an activation set $A_i$. Each element $\alpha_{i,j} \in A_i \subset \mathbb{N}^+$ is the index of a local maximum of $P_i$. In other words, each element $\alpha_{i,j} \in A_i$ with $j \in \{1, \ldots, |A_i|\}$ denotes a framenumber where the poselet score was locally maximal. We call poses depicted in frames indexed by $A_i$ *support poses*, as they will support the discovery of key-poses.

The temporal interval $[\alpha_{i,j-1}, \alpha_{i,j+\mathbb{1}_s}]$ covers one complete stroke for a good poselet. Here, $\mathbb{1}_s$ is an indicator function:

$$\mathbb{1}_s = \begin{cases} 1 & \text{if s} \in \{\text{freestyle, backstroke}\} \\ 0 & \text{else.} \end{cases} \tag{3.17}$$

The usage of the indicator function is necessary because of the aforementioned anti-symmetry of freestyle and backstroke motions. With the poselets not being able to distinguish left and right body side, we usually observe two distinct maximal peaks

within one stroke. The length of a stroke is given by $\alpha_{i,j+1} - \alpha_{i,j-1}$ for freestyle and backstroke and $\alpha_{i,j} - \alpha_{i,j-1}$ for breaststroke and butterfly swimmers.

We use a simple heuristic to filter potential false activations, i.e., activations that do not represent a support pose. Therefore, we can build a histogram for all stroke intervals in $A_i$ and determine the dominant stroke frequency $f_{stroke}$ for a swimmer via the maximal value in the histogram. False activations can be iteratively discarded by greedily deleting $\alpha_{i,j} \in A_i$ that produce frequencies much smaller than $f_{stroke}$. An interval $[\alpha_{i,j-1}, \alpha_{i,j+\mathbb{1}_s}]$ are called a regular interval if $\alpha_{i,j+\mathbb{1}_s} - \alpha_{i,j-1} - f_{stroke} < \lambda$ holds for a small $\lambda$. In the experiments, $\lambda = 0.1 \cdot f_{stroke}$ proved to be a decent choice.

### Activation Likeness

Not all poselets are informative for predicting key-poses. We propose to measure the self-similarity of an activation set in order to sort sorting activation sets according to their goodness. Adjacent groups of activations should be very similar if the series is regular. Dissimilarity indicates that the series is irregular due to false positive or missed activations.

Let two vectors $\mathbf{a}, \mathbf{b}$ of adjacent poselet activations in activation set $A_i$ be defined as

$$\mathbf{a} = [\alpha_{i,j}, \alpha_{i,j+1}, \alpha_{i,j+2}]^T, \mathbf{b} = [\alpha_{i,j+3}, \alpha_{i,j+4}, \alpha_{i,j+5}]^T \tag{3.18}$$

We reuse our semimetric derived for configurations to obtain a measure of activation likeness. Subtracting the element-wise mean from both vectors, normalizing such that $\mathbf{a}^T\mathbf{a} = \mathbf{b}^T\mathbf{b} = 1$ and enforcing a positive scaling factor, we use Equation 3.9 to compute the distance between $\mathbf{a}$ and $\mathbf{b}$. All poselet activation series can then be sorted according to the resulting distances, with smaller distances indicating more regularity. Note that the activation likeness can change within the same video, e.g., if the flow velocity and the posture of the swimmer changes, some poselets might stop working while others capture the changed situation better. We evaluate the likeness of all poselets continuously, using only a subset of poselets with the most regular activations to predict key-poses. In the experimental section, the effect of using different sets of poselet activations will be evaluated thoroughly.

### Key-pose Estimation

Our key-pose estimation scheme builds on support-poses, defined by regular poselet activation sets $A_i$. We model the occurence of a key-pose $\kappa$ (= number of key-frame) within a motion cycle by a maximum a posteriori probability

$$\kappa^* = \arg\max_{\kappa} p(\kappa|A_\kappa). \tag{3.19}$$

The $\kappa$-specific activation set $A_\kappa \subset \bigcup_i A_i$ is a subset of all activation sets. This set holds all regular activation interval borders $\alpha_{i,j}$ and $\alpha_{i,j+1+\mathbb{1}_s}$ such that $\alpha_{i,j} < \kappa < \alpha_{i,j+1+\mathbb{1}_s}$. Including only the closest support poses follows the notion that only temporally close support poses are decent predictors for a key-pose. We can rewrite Equation 3.19 by means of Bayes' theorem, yielding

$$\kappa^* = \arg\max_{\kappa} p(A_\kappa|\kappa)p(\kappa). \tag{3.20}$$

We may assume a uniformly distributed prior for now and only look into the maximum likelihood of Equation 3.20. By making the simplifying assumption that activations from different poselets are independent from each other given $\kappa$, this leaves us with the log-likelihood

$$\begin{aligned} \kappa^* &= \arg\max_{\kappa} \prod_i p(\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}|\kappa) \\ &= \arg\max_{\kappa} \sum_i \log p(\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}|\kappa) \end{aligned} \tag{3.21}$$

Vividly, each poselet gives its own estimate of where a key-pose occurs in the time series. The final estimate for $\kappa^*$ is then the sum over different contributions $p(\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}|\kappa)$. Each probability $p(\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}|\kappa)$ is modeled by a Normal distribution as follows.

### Key-pose Probabilities

A problem when predicting key-pose occurrences is the stroke frequency between different athletes. In our scenario, different swimmers spend different time executing a cycle. Even for the same athlete, different current velocities yield different stroke timings. Let us define the length of a stroke interval as $c_i = \alpha_{i,j+1+\mathbb{1}_s} - \alpha_{i,j}$. We

can use $c$ to normalize a key-pose occurrence given a poselet activation interval $[\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}]$ by

$$\hat{\kappa}_i = \frac{\kappa - \alpha_{i,j}}{c_i} \tag{3.22}$$

It follows that $\hat{\kappa}_i \in [0, 1]$. Using a database of training videos with annotated key-poses $\kappa$, we apply our poselet detector and extract activation intervals as described above. For each regular activation interval that surrounds a key-pose annotation $\kappa$ in a training video, we apply Equation 3.22 and retrieve $\hat{\kappa}_i$. We then fit a normal distribution $\mathcal{N}(\kappa; \mu_i, \sigma_i^2)$ to all normalized key-poses $\hat{\kappa}_i$ in the training set. For inference, each estimated Gaussian is linearly transformed to approximate the likelihood $p(\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}|\kappa)$ in Equation 3.21 by

$$\begin{aligned} \kappa^* &= \arg\max_{\kappa} \sum_i \log p(\alpha_{i,j}, \alpha_{i,j+1+\mathbb{1}_s}|\kappa) \\ &= \arg\max_{\kappa} \sum_i \log \left( c_i \cdot \mathcal{N}(\kappa; \mu_i, \sigma_i^2) + \alpha_{i,j} \right) \\ &= \arg\max_{\kappa} \sum_i \log \mathcal{N}(\kappa; c_i\mu_i + \alpha_{i,j}, c_i^2\sigma_i^2) \end{aligned} \tag{3.23}$$

The formulation above assumes that the prior probability $p(\kappa)$ is uniformly distributed, thus reducing the maximum a posteriori (MAP) estimate to a maximum likelihood (ML) estimate. The complete Bayesian view, however, assumes that a prior $p(\kappa)$ is used. Finding a good prior initialization can be challenging. For instance, an expert could annotate just one single occurrence of a key-pose for a specific video manually. This single annotation can be propagated to all other cycles by designing a normal distribution similar to the likelihood model, setting the standard deviation $\sigma_i^2$ to a fixed small value. We will elaborate and evaluate the idea of ignoring or setting a prior in the experimental section.

## 3.3 Experiments

The performance of the proposed key-pose estimators are validated and discussed on a subset of the *mmc-swim-channel* dataset covering 30 swimmer videos of different freestyle swimmers (six male, eight female, ages 15 to 25, different body sizes) in the Leipzig swimming channel. A human expert annotated all frames that depict one of four key-poses (illustrated in Figure 3.5), overall 1696 occurrences. Note that

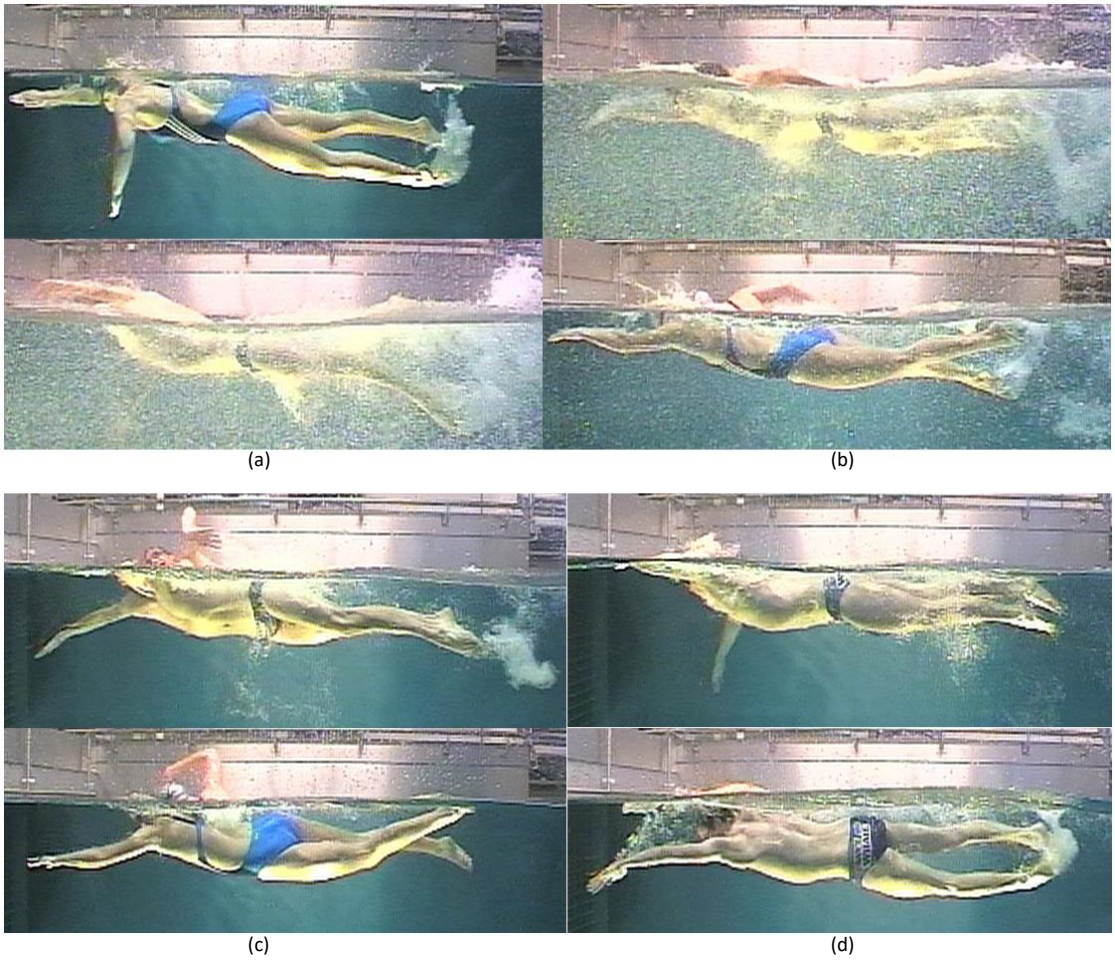



Figure 3.5: All four key-poses evaluated in this chapter. A key-pose for anti-symmetrical swimmers occurs on the left (depicted on top) and on the right (depicted on bottom) half of the body. (a) Upper arm under water, 270 degrees angle in the camera image. (b) Hand just leaving the water surface. (c) Upper arm 90 degrees relative to the image plane, above the water surface. (d) Hand touching water surface.

these key-poses annotations are not the same as in Table 2.1, where key-poses where re-annotated at a later time for the larger dataset.

A mixture model of 16 poselets (one root + 15 arm poselets) was trained from 1200 distinctive images with given pose annotations. The root detector covers the whole swimmer and gives a root hypothesis for spatially max-pooling all other poselet in the mixture. The rest of the model covers 15 temporally clustered arm configurations (shoulder, elbow, and wrist) from all arms in the dataset, i.e., from the left

and right body side. The estimation of key-pose occurrences is evaluated in 30-fold cross-validation, where key-pose estimators are trained on all subsets of 29 videos and evaluated on the remaining video. The final score is averaged over all folds.

### Performance Measure

In general, we distinguish between different types of detections. An estimated key-pose is counted as a true positive $TP$ if it was estimated within $\pm$ 10 frames of a ground truth label an no other estimate is closer to the same ground truth. Otherwise, the estimated pose is counted as a false positive $FP$. A ground truth key-frame without a matched prediction is a false negative $FN$. The performance of the proposed algorithm is evaluated using the Percentage-of-Correct-Key-Frames measure (PCKF). This measure normalizes the deviation between a predicted key-frame and the ground truth with the stroke length. The normalized deviation threshold $\delta$ on the x-axis is then compared against the percentage of correctly identified key-frames on the y-axis. Normalizing the deviation with the length of the stroke is natural as a deviation of two frames for a fast-moving swimmer produces a larger error than two frames deviation for a slow swimmer. All PCKF curves will be compared at a deviation threshold of $\delta = 0.03$, denoted $PCKF@0.03$, which reflects the mean human error tolerance between different experts. A threshold of 3% is approximately equivalent to a deviation of $\delta \leq \pm 2$ frames averaged over different videos with different water flow velocities and different annotators. Additionally, we also report $PCKF@0.02$ to give a better intuition for the PCKF curve. As the PCKF measure makes no statement about false positive detections, we additionally consult the precision Pr of the model, defined as

$$\mathrm{Pr} = \frac{TP}{TP + FP} \tag{3.24}$$

The precision is an indicator of how many false guesses the predictor made and will be presented with each PCKF curve. A complete overview of the achieved scores is given in Table 3.1.

### Comparison between ML, Prior and MAP predictions

We first compare the performance of different probabilistic approaches for the prediction of key-pose one in Figure 3.6 (left). Our maximum likelihood (ML) approach

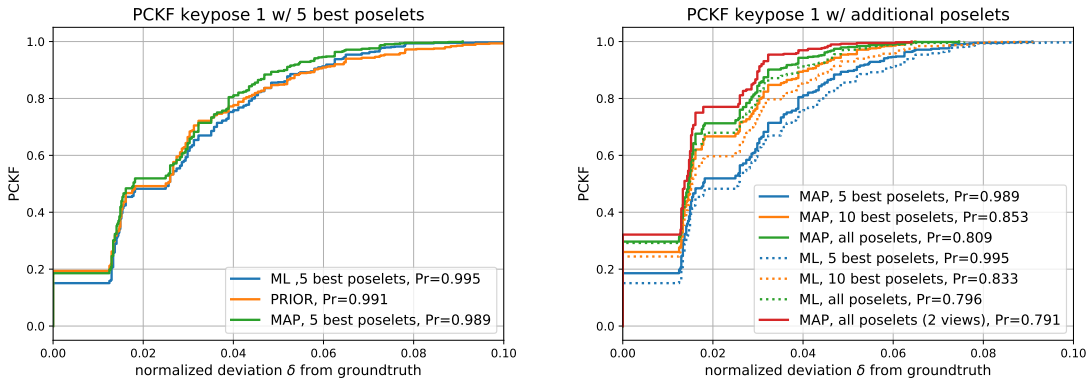



Figure 3.6: Left: PCKF for maximum likelihood (ML), prior and maximum a posteriori (MAP) estimates for occurrences of key-pose 1. Right: MAP and ML estimates considering a different number of activation sets. While the PCKF steadily increases, the precision decreases with the introduction of false positives. The red curve represents the best model which includes a second camera view.

only estimates the occurrence of a key-pose based on poselet activation sets. We also evaluate a prior probability as well as the combined MAP probability.

Using the time series of the five best-performing poselets, the maximum likelihood model achieves a PCKF of 61.6% at an error threshold of $\delta = 0.03$ with a precision of $Pr = 0.995$. The large precision indicates that only a small number of false positive key-pose predictions were made. A prior estimate can be included by using just a single expert annotation of a key-pose in one motion cycle, assuming that the temporal occurrence will be similar in all other cycles. The prior can be incorporated into the whole model only if we are able to express it as a normal distribution. Therefore, we use Equation 3.22 to normalize a single expert key-pose annotation per video, given the activation sets of the detector. From this annotation, the mean is naturally given by the index of the key-frame itself. As we are not able to determine an uncertainty $\sigma^2$ from just one example, a tight value reflecting the belief that an expert annotation is very precise is chosen empirically by setting $\sigma^2 = (0.04 \cdot f_{stroke})^2$. Applying only the prior estimate to all videos leads to surprisingly good key-pose estimates, yielding a PCKF of 0.641 at $\delta = 0.03$ with precision 0.990. If the prior is incorporated with the ML estimate, forming a complete Bayesian MAP hypothesis, we find that its resulting PCKF curve (green) in Figure 3.6 slightly surpasses both other probability estimates.

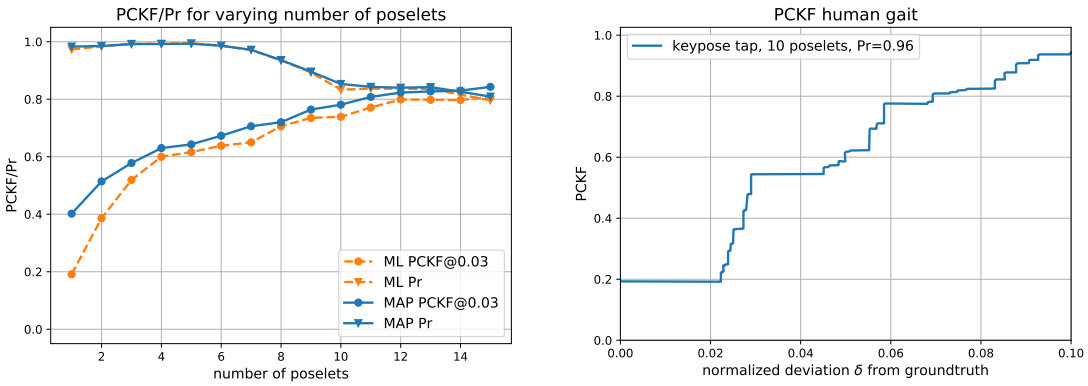



Figure 3.7: Left: PCKF@0.03 for a varying number of poselets. Right: PCKF for walking persons in the CMU Motion of Body database [47] for human gait analysis.

An interesting difference in key-pose estimation performance can be seen when explicitly distinguishing between left and right body side. As the detectors are not able to make this distinction, we manually evaluated the PCKF for the left side (MAP5_left) and the right body side (MAP5_right) in Table 3.1. We find that the PCKF for the left is considerably higher than the score for the right. This effect can be attributed to the poselet confidence. While poselets may not be able to distinguish explicitly between left and right body side, detecting the part of the body that is further away from the camera is more difficult due to more noise and considerable foreshortening, consequently leading to significantly smaller poselet confidences. This effect constitutes irrecoverable support poses and therefore inferior key-pose estimation scores for the right body side.

### Increasing Number of Poselets

The performance of spatiotemporal poselet detectors can be enhanced by incrementally adding additional, subpar poselet activations to the MAP estimation. Results of this process are depicted in Figure 3.6 (right) and specifically for a PCKF@0.03 in Figure 3.7 (left). The overall performance clearly improves up to a PCKF of 0.85 when using all poselets and a prior, independent of whether the poselets are eligible. On the other hand, the precision drops significantly to $Pr = 0.809$ as an unwanted side effect. Most poselet activation sets, even the bad ones, contain at least some valid stroke intervals which improve the performance. However, they also

Figure 3.8: Examples from an additional camera view which can be used to improve predictions for key-pose occurrences.

contain stroke intervals which do not represent a support-pose and introduce false predictions.

Figure 3.7 (left) illustrates that if a prior can be added to the model, the PCKF is consistently higher than when using only a poselet detector for key-pose prediction. Especially for very few poselets, it becomes apparent that adding an additional Gaussian in form of a prior to Equation 3.20 is beneficial to the overall score. However, in terms of a fully automatic approach to activity kinematics estimation, adding a prior might not be desirable, leaving us with the ML scores as a baseline.

We obtain our best ML results with high precision by leveraging the activation likeness of a poselet, as described in Section 3.2.5. We empirically set the likeness threshold $\lambda = 0.005$ and use all poselets where the activations of a cycle compared with the next cycle produce a likeness score smaller than $\lambda$. The results are illustrated in Table 3.1 and dubbed ML-AL. While the PCKF scores lie above the average ML10 model, we did not produce too many false positive detections.

### Additional Camera View

The results from the last experiment - that key-pose estimates can be improved with additional and potentially bad poselet activation sets - inspired the following experiment. An additional mixture of poselets, one athlete detector plus six arm poselets, is trained for a second synchronized camera view that captures each swimmer from above the swimming channel. Examples for this camera view are depicted in Figure 3.8. While this camera is not able to monitor any movement below the water surface due to froth, it detects the swimmer and any arm movement above the water surface quite well. Note that this model behaves like a model trained for

| | PCKF@0.02 | PCKF@0.03 | precision |
|---|---|---|---|
| ML5 | 0.482 | 0.616 | 0.995 |
| ML5_left | 0.524 | 0.64 | - |
| ML5_right | 0.422 | 0.48 | - |
| ML10 | 0.596 | 0.739 | 0.833 |
| ML15 | 0.679 | 0.808 | 0.796 |
| ML-AL | **0.625** | **0.751** | 0.985 |
| MAP5 | 0.518 | 0.643 | 0.993 |
| MAP10 | 0.666 | 0.781 | 0.853 |
| MAP15 | 0.712 | 0.843 | 0.809 |
| prior | 0.491 | 0.665 | 0.993 |
| MAP18 (2 views) | **0.770** | **0.895** | 0.791 |

Table 3.1: Scores for the proposed key-pose detector for maximum likelihood (ML), prior and maximum a posteriori (MAP) key-pose estimation using a different number of considered poselets (5,10,15,18). ML-AL uses activation likeness to determine the best performing poselets within a time window using the likeness threshold $\lambda = 0.005$.

a symmetrical swimming style. Poselet activations only occur once a cycle because each arm is detected by its own set of poselets. Hence, we mirror poselets detecting the same arm configuration on the left an right body side and join them together in one activation set. The resulting three activation sets are joined with the other 15 from the main view. Including the second view improves the PCKF by another 4.5% to an overall score of 0.895, with only a minor drop of 0.01 in precision. This resulting PCKF curve is also depicted in Figure 3.6 (right, red graph). We will use this model as a baseline for the comparison in the next chapter.

### Human Gait Analysis

Although the key-pose estimation scheme presented in this chapter was initially developed for swimmers depicted from a side view, it is not necessarily restricted to this application. In order to show that this approach can be transferred to other cyclic motion, an additional experiment was carried out on the CMU Motion of Body database [47], which was commonly used for human gait analysis in the past. We trained a ten-part poselet model for leg configurations of ten walking persons depicted from the left side (Figure 3.9). A maximum likelihood estimator was trained for the key-pose depicting a person when his or her left or right heel touches the ground at the end of swing phase preluding the stance phase. The model was then

Figure 3.9: Examples from the CMU Motion of Body database.

evaluated on a different set of ten videos with the same evaluation criterion as for the swimmers. The result is depicted in Figure 3.7 (right). As the walking frequency is usually larger than the stroke-frequency of a swimmer, the allowed deviation of $\pm 2$ half-frames for swimmers roughly translates to 8% deviation for walkers. Within this threshold, we achieve a PCKF of 0.801 with a precision of 0.962.

## 3.4 Summary

The support pose detector and key-pose estimation scheme established in this chapter structurally follows the formulation of a deformable part model [34] for object detection with some important modifications:

- Instead of just detecting spatial pose configurations, the model is built for spatiotemporal pose detection. To that, multiple poselets for the same part of the body at different temporal stages of cyclic motion are evaluated simultaneously and continuously.

- A semimetric for clustering poselets measures the similarity of poses and clusters pose configurations that are spatially and temporally close.

- We relax the constraint that the covariance of the spatial relationship between two poselets has to be diagonal and allow for arbitrary covariance matrices, which we use to define activation regions for the poselet ensemble.

- The proposed poselet ensemble is evaluated hierarchically, i.e., the activation of a poselet in the image is conditioned on the score of a top-level classifier.

- A simple averaging scheme is proposed for identifying latent key-poses only from the best performing poselets in the model.

The HoG feature representation used for training poselets is outdated nowadays. We will show in the following chapters that novel pose estimation systems based on DCNN features can significantly outperform key-pose estimation results reported in this chapter. Nevertheless, key-pose estimation via support poses identification is neither obsolete nor superfluous. Beyond the key-poses depicted in 3.5, coaches are sometimes interested in detecting more intricate pose features. We found that pose features can sometimes be so inconspicuous that even deeply learned image features are not able to identify them directly. This is where the strength of a spatiotemporal poselet detector lies. Even if we are not able to detect a latent pose feature directly, it can be inferred indirectly given that we are able to determine stable support poses.

# 4 — Athlete Pose Estimation with Deep Neural Networks

Prior to the revival of deep convolutional neural networks by Krizhevsky et al. [66], the estimation of all major joints of the swimmer pose with HoG-based [20] feature representations only yielded unsatisfactory results. Over the past few years, handcrafted features such as HoG have been replaced by deeply learned features, resulting in significant performance improvements for various computer vision tasks.

In this chapter, we propose a pose estimator for swimmers that builds on the basic pictorial structure model briefly introduced in Chapter 2. The estimator builds on a deep neural network to learn appearance representations for all joints of the human body. The output of this network is passed to a computational graph [4, 60], which implements the classical deformable structure model[1]. We also include image dependent pairwise relation (IDPR) terms. The notion behind these terms is that the visual context of a part can be used to make a prediction for the location of neighboring parts and therefore improve part location estimates in the whole framework (Figure 4.1). We show that IDPR terms can be computed efficiently as part of the joint appearance. In the experimental section, the proposed pose estimation model, which we denote CG-IDPR, is compared with the system introduced by Wei et al. [106]. They propose a convolutional network for pose estimation that uses several fully convolutional blocks for implicitly learning a representation for the structure of human poses.

The direct estimation of the human pose allows for predicting key-poses directly by detecting their associated pose features. Therefore, we evaluate two distinct methods. Method one simply predicts key-poses by querying pose features, method two infers key-poses via a random forests classifier.

[1]The proposed pose estimator was published in [112].

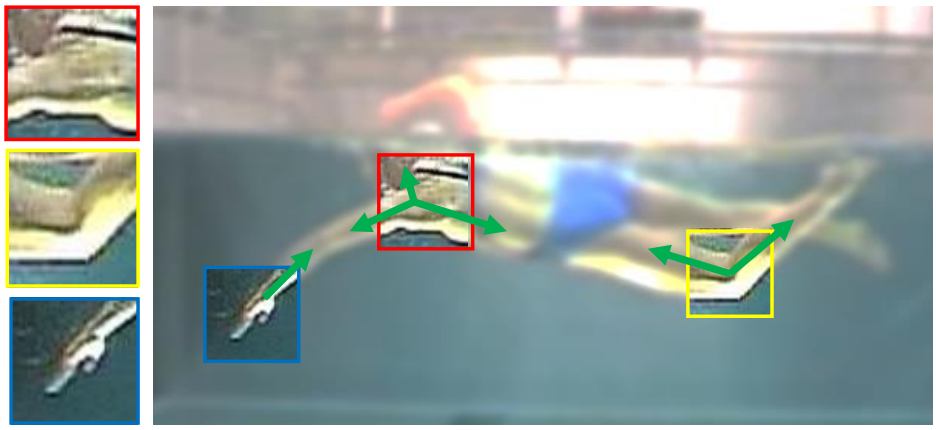


Figure 4.1: Image dependent pairwise relation terms model the notion that a local image patch around a joint (left) itself is a strong indicator of where the neighboring joints are located (implied by green arrows on the right).

## 4.1 Related Work

With the resounding success of deep convolutional neural networks, hand-crafted models for the appearance and the deformable structure of a pose have been replaced by neural networks recently [100, 106]. The joint training of pictorial structures and DCNNs has been picked up by Tompson et al. in [97] who combine a DCNN with a Markov Random Field and successfully show that their model can successfully exploit geometric relationships between body joint locations. Chen and Yuille show in [15] that local appearance of parts learned by a DCNN can also be used to predict neighboring part locations and thereby improve the prediction performance of the whole model. Yang et al. [109] formulated DPMs in the context of a DCNN by introducing a message passing layer which recurrently performs inference on part detection maps. Compared to our system, their formulation only covers a basic DPM formulation excluding image dependent pairwise relations.

Over the last years, more and more pose estimation approaches have replaced explicit deformation modeling with a deep learning scheme [16, 51, 78, 95, 108, 117]. This chapter specifically discusses the work of Wei et al. [106], who introduce simple five-layered convolutional blocks to improve the joint detections of an appearance network using an advised training procedure. While this method might not produce the latest state-of-the-art results [65, 95], it is charmingly simple and easy to adapt to novel applications.

## 4.2 Pose Estimation with Image Dependent Relation Terms

### 4.2.1 Base Model

The goal of articulated human pose estimation is to find a hypothesis $L$ for all joints given some image evidence. To that goal, we define the following model energy function. The goodness of a placement $L = \{\mathbf{l}_1, ..., \mathbf{l}_{|V|}\}$ of all joint parts for an image $I$ can be evaluated by

$$\begin{aligned} F(L, T, I; \theta, \mathbf{w}) \; &= \overbrace{\sum_{i \in V} w_i \Gamma_i(\mathbf{l}_i)}^{appearance} + \overbrace{\sum_{(i,j) \in E} \Psi(\mathbf{l}_i, \mathbf{l}_j, t_{ji}, \mathbf{w}_{\mathbf{ji}}^{\mathbf{t_{ji}}})}^{structure} \\ &+ \underbrace{\sum_{(i,j) \in E} w_{ij} \Lambda_i(\mathbf{l}_i, t_{ij}) + \sum_{(i,j) \in E} w_{ji} \Lambda_j(\mathbf{l}_j, t_{ji})}_{IDPR} \end{aligned} \tag{4.1}$$

where $\theta$ are the parameters of the DCNN, $T = \{T_{ij}, T_{ji} : (i,j) \in E\}$ are spatial relationships (=clustered offsets between pairs of neighboring parts) and $\mathbf{w}, w$ are learned weights balancing the different terms.

Let us for the moment ignore the detailed mathematical notation and just discuss the three semantically distinguishable terms in this equation. The first sum in equation 4.1 evaluates the appearance of a specific placement of the parts. This term should be large if a joint $i$ is present at a position $\mathbf{l}_i$ in the image and small otherwise. The second sum represents the structure model and rates the goodness of placements of neighboring parts. Ideally, this term is large if we find two neighboring parts in an optimal configuration and it should be small for unrealistic part placements. Both terms describe the traditional energy formulation for matching a deformable part model according to [35] as briefly discussed in Chapter 2. The last two sums are summarized under the term IDPR and assess the local belief that a neighboring part is placed at a certain location based on image evidence. Given a set of model appearance parameters $\theta$, learned weights $\mathbf{w}, w$ and an image $I$, the goal of an inference algorithm is to maximize $F(.)$ in order to find the best fitting joint hypothesis $L$. In the following, we will discuss the mathematical terms in Equation 4.1 in detail.

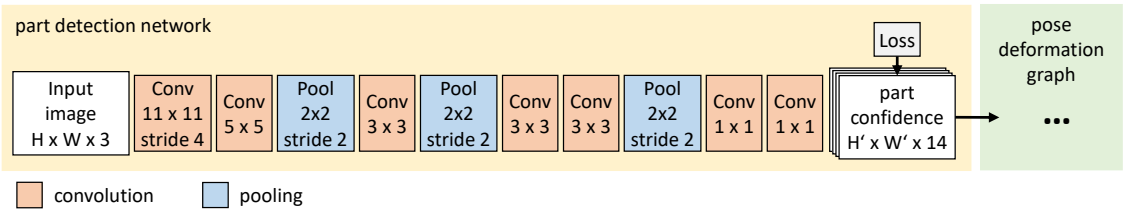



Figure 4.2: The part detection network is based on the AlexNet structure [66] with the last fully connected layers transformed to convolutional kernels as proposed by [91]. The network returns confidence maps for joints in very specific neighborhood configurations which are passed to the pose deformation graph for further processing (see Section 4.2.2).

### Part Appearance

As the depiction of the human pose in an image can be highly articulate, we allow for multiple offsets between two joints and model a set of spatial relationships $T_{ij} = \{1, ..., K\}$ and $\mathbf{r}_{ij} = \{\mathbf{r}_{ij}^{t_{ij}} : (i,j) \in E \wedge t_{ij} \in T_{ij}\}$, where the offsets between two neighboring parts $i$ and $j$ in a training set are clustered into $K$ clusters with cluster identifiers $t_{ij} \in T_{ij}$ and cluster centers $\mathbf{r}_{ij}^{t_{ij}}$. Given a set of swimmer images with a ground truth joint annotations $\mathbf{l}_i$ with $i \in \{1, ..., |V|\}$, we extract image patches $I(\mathbf{l}_i)$ centered around each joint $i$. A class label out of a set of labels $\mathcal{L}$ is assigned to each patch $I(\mathbf{l}_i)$, consisting of its joint class $c = i$ and identifiers $\mathbf{t}_{i\mathcal{N}_i}$ obtained from clustering each offset between a joint $i$ and its neighbor $j \in \mathcal{N}_i$ in the training data. Intuitively, all part patches sharing the same part label depict the same joint with all the neighboring joints having the same spatial configuration in the image. This procedure leads to a rather large set of $|\mathcal{L}| = \sum_{i \in V} \prod_{j \in \mathcal{N}_i} |T_{ij}| + 1$ specialized classes[2] for detecting very specific configurations of joints and one class representing background patches.

We use a deep neural architecture based on an AlexNet [66] with a soft-max loss to predict the class labels for all extracted joint patches. The architecture for this network is depicted in Figure 4.2. After training, we implement the trick presented in [91] and transform the last two fully connected layers to $1x1$ convolutional kernels. This allows for feeding an image of arbitrary size into the network instead of single

[2]For a human pose with 13 joints (no neck) and $K = 11$ clusters per joint pairing, $|\mathcal{L}| = 3554$. Including a joint annotation for the neck, $|\mathcal{L}| = 4775$ classes of joint appearances have to be predicted.

image patches. As a result, we receive $|\mathcal{L}|$ confidence maps, one for each patch class label. Each value in an output map represents the confidence for an associated, patch sized input window in the input image. From the specific output maps of the joint detection network, general joint detectors for joint $i$ can be derived by element-wise summing over all network output maps where the patch class label incorporates the joint class $c = i$. We denote the score of an appearance detector for joint $i$ at location $\mathbf{l}_i$ with $\Gamma_i(\mathbf{l}_i)$ in the following. The sum over all appearance detectors constitutes the appearance term in Equation 4.1.

### Image Dependent Pairwise Relation Terms

Our model formulation incorporates image dependent pairwise relation (IDPR) terms. Therefore, we define an image dependent term $\Lambda_i(\mathbf{l}_i, t_{ij} = k)$ specifically tailored to represent a joint $c = i$ at location $\mathbf{l}_i$ in a particular neighbor configuration $k$ with neighbor $j$. A score for $\Lambda_i$ can be obtained by summing over all class outputs of the appearance network from the previous section where the patch class label incorporates joint $c = i$ and additionally the specific cluster id $t_{ij} = k$ for the edge $(i, j) \in E$. The sum over all IDPR terms for all neighboring joints constitutes the IDPR part of objective Equation 4.1.

### Deformation Terms

The classical deformation terms assess the fit of two detections for neighboring parts and allow for some flexibility in the relative positions between both. In our model formulation, it is defined by

$$\Psi(\mathbf{l_i}, \mathbf{l_j}, t_{ji}, \mathbf{w}_{ji}^{t_{ji}}) = \left\langle \mathbf{w}_{ji}^{t_{ji}}, \delta\left(\mathbf{l_i} - \left(\mathbf{l_j} + \mathbf{r}_{ji}^{t_{ji}}, \right)\right)\right\rangle \tag{4.2}$$

where $\delta(\Delta\mathbf{l}) = [\Delta x, \Delta x^2, \Delta y, \Delta y^2]^T$ and $\Delta\mathbf{l} = [\Delta x, \Delta y]^T$ is a deformation feature and $\mathbf{w}_{ji}^{t_{ij}}$ are learned deformation weights penalizing larger magnitudes of $\Delta\mathbf{l}$. The notion behind deformation terms is to allow for a part $j$ to deviate slightly from its ideal placement $\mathbf{r}_{ji}^{t_{ij}}$ relative to part $i$. While small deviations from the ideal relative offset between two parts lower the score only by a small margin, there will be a larger penalty for a part that is located further away from the ideal offset. The steepness of this score decay is defined by the deformation weights $\mathbf{w}_{ji}^{t_{ji}}$, where smaller deformation weights allow for a larger deviation from the optimal displacement.

Felzenszwalb et al. [33] propose to use a generalized distance transform [36] to implement the process of matching parts in a deformable configuration efficiently. The transform is applied on the appearance map of one joint and spreads high scoring detections of a part to nearby locations. The spread distance is restricted by the quadratic deformation parameters. After shifting the map by the ideal offset between the two joints, the spread high scores can positively influence the final score sum between both parts.

### Weight Parameter Learning

The energy formulation in Equation 4.1 includes weight terms $w_i, w_{ij}$ and $w_{ji}$ for appearance and image dependent pairwise relations, respectively, as well as weights $\mathbf{w}_{\mathbf{ji}}^{\mathbf{t_{ji}}}$ influencing the allowed deformation between two neighboring parts. We learn all weight parameters by defining sparse feature vectors $\phi(\mathbf{I^n}, \mathbf{l^n}, \mathbf{t^n})$ representing the concatenation of image dependent terms, IDPR terms, deformation features and a bias term set to 1 in image $\mathbf{I^n}$ with annotated part locations $\mathbf{l^n}$ and derived type labels $\mathbf{t^n}$ for training example $n$. Note that sparsity of $\phi(\mathbf{I^n}, \mathbf{l^n}, \mathbf{t^n})$ follows from the fact that all dimensions in this feature that do not correspond to type labels and deformation terms observed in image $\mathbf{I^n}$ are set to zero. We collect a training set of positive examples from ground truth joint annotations and negative examples mined through hard-negative mining detections on images depicting no athletes. Finally, all weights $\mathbf{w}$ are learned by training a linear support vector machine on positive and negative training examples as described in [15].

### Inference

The maximization of equation 4.1 yields a set of joint placements $L^*$ and their pairwise relations $T^*$:

$$L^*, T^* = \text{argmax}_{L,T} F(L, T, I; \theta, \mathbf{w}) \tag{4.3}$$

Optimization of this energy function defined over a tree-shaped pose graph $G$ is achieved by means of dynamic programming using the max-sum algorithm, which allows for recursively splitting the part matching problem into subproblems. Let $S_i(\mathbf{l}_i)$ denote the model score for a sub-tree $G_i$ of $G$. Starting with the leafs of the human pose tree, i.e., wrists and ankles, the placement for each joint is simply given

by the appearance maps. Continuing from the leaves, a recursive score for each parent nodes is given by

$$S_i(\mathbf{l_i}) = w_i \Gamma_i(\mathbf{l}_i) + \sum_{n \in \mathcal{C}_i} s_{in} \tag{4.4}$$

with

$$\begin{aligned} s_{in} = \max_{l_n, t_{ni}} (\Psi(\mathbf{l_i}, \mathbf{l_n}, t_{ni}, \mathbf{w_{ni}^{t_{ni}}}) + w_{ij} \Lambda_i(\mathbf{l}_i, t_{in}) \\ + w_{ji} \Lambda_n(\mathbf{l}_n, t_{ni}) + S_n(\mathbf{l}_n)). \end{aligned} \tag{4.5}$$

$S_i$ equals the appearance confidence of the parent joint plus the score of the children after deformation terms and IDPR terms have been evaluated. Determining the final pose hypothesis involves the technique of backtracking. A global maximum of the energy function is found by picking the root location $\mathbf{l_1^*} = \text{argmax}_{\mathbf{l}}(S_1(\mathbf{l}) > \tau)$ given a detection threshold $\tau$. The location of all other parts can be traced back recursively. Therefore, for all children of a vertice, we determine the location offset $\mathbf{r}_{ij}^{t_{ij}}$ of the part detection that contributed the most to the best score. In order to perform the backtracking procedure efficiently, it is beneficial to trace joint offset variables in lookup tables during the forward pass of an image.

### 4.2.2 Pictorial Structures as Computational Graphs

We implement the model described in the previous section as a computational graph. Therefore, we first focus on a single step for pairwise optimization of a sub-tree as described by Equations 4.4 and 4.5. Afterward, we present the graph structure for predicting the complete pose of an athlete.

#### Pairwise Joint Optimization

The computational graph depicted in Figure 4.3 specifically solves one step in the max-sum algorithm as illustrated in Equations 4.4 and 4.5. In this figure, rectangles correspond to confidence maps of appearance and IDPR terms, circles denote an operation on maps and arrows visualize the data flow through the graph. We assume in this depiction that the edge between a child part $j$ and its parent part $i$ has been clustered $K$ times. As a result, there are $K$ different IDPR maps describing the belief for the position of parent $i$ seen from part $j$ and another $K$ IDPR maps describing

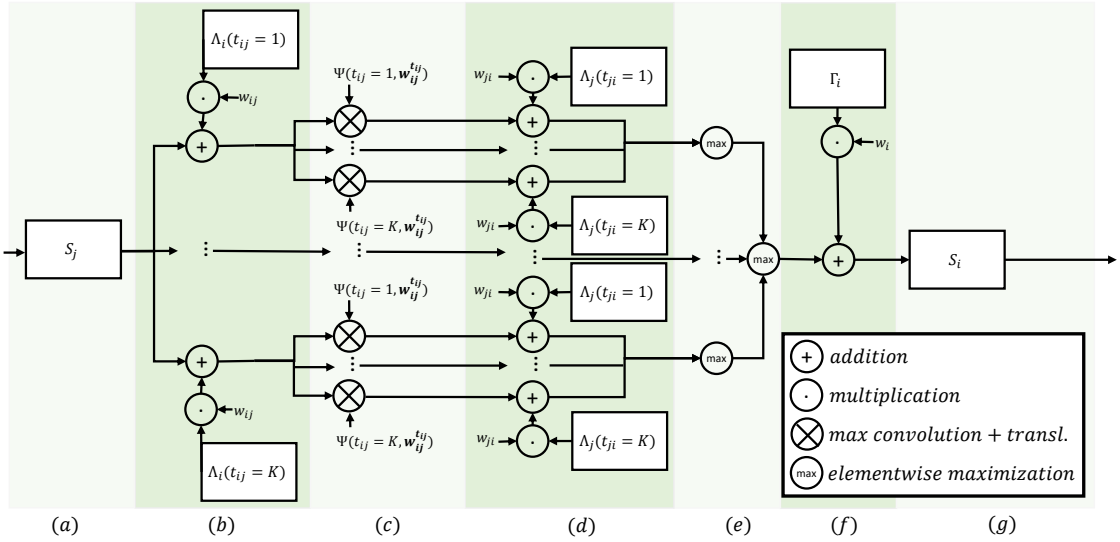



Figure 4.3: One step of the dynamic program discussed in Section 4.2.1. Partial solutions (a) of a child joint is added to the weighted IDPR terms for the child node in (b). Distance transformations are performed in (c) for different pairwise relations. In (d), the transformed results are added to parent joint IDPR maps. The elementwise maximization in (e) yields the best child joint scores, which are added to the parent joint appearance map in (f), which yields the results of Equation 4.4 in (g). A high scoring joint configuration can be obtained by backtracking the optimal path in this data flow graph.

the belief for the position of child $j$ seen from parent part $i$. Also, $K$ clusters implicitly define $K$ different deformation weights $\mathbf{w}$ together with their offsets $\mathbf{r}_{ji}$ between part $j$ and $i$. A solution $S_i$ for the sub-tree $G_i$ is computed as follows. The sub-tree scores in Figure 4.3 (a) is added to the weighted IDPR terms in (b) of part $j$. This step is executed $K$ times for all $K$ IDPR maps. For each sum, an operation called a max-convolution is performed, yielding overall $K^2$ transformed maps in stage (c). We will introduce the max-convolution in detail in the next section. Each set of transformed confidence map is now added to the IDPR terms of the parent in (d). Note that stages (b) to (d) compute the Cartesian combinations of all IDPR terms between two parts. In step (e), the element-wise maximum of all intermediate results are combined, yielding the best confidence scores from all possible locations of the child parts. The result is then added to the appearance of the parent, yielding the final score map $S_i$. We denote the complete optimization graph in Figure 4.3 a *part-to-part layer*.

**Max-Convolution.** While the operations of addition, multiplication and element-wise maximization are straight forward to implement, the formulation of a max-convolution has to be defined more precisely. It implements the classical deformation computation, for which [33] initially proposed a generalized distance transform, efficiently as a filtering operation.

Let $H$ be a regular grid and $f : H \to \mathbb{R}$ be a function defined on that grid. In our case, $f(.)$ is a confidence map. The generalized distance transform $D_f(p)$ at a point $\mathbf{p}$ on $H$ is defined over neighboring points $\mathbf{q}$ given a kernel $h(.)$ as

$$D_f(\mathbf{p}) = (f \otimes h)(\mathbf{p}) = \max_{\mathbf{q}}(f(\mathbf{q}) - h(\mathbf{p} - \mathbf{q})) \tag{4.6}$$

The operator $\otimes$ denotes the max-convolution. Note that this formulation of the distance transform somewhat resembles a discrete convolution

$$(f \star h)[\mathbf{p}] = \sum_{\mathbf{q}} f[\mathbf{p}] \cdot h[\mathbf{p} - \mathbf{q}] \tag{4.7}$$

where the sum is replaced by the max operator and the product is replaced by a summation. This similarity inspires the name max-convolution, which in contrast to the standard convolution is highly non-linear. Classical graph-based pose estimation approaches often assume that the deformation costs are modeled by quadratic terms, i.e., $h(\mathbf{p}-\mathbf{q}) = \langle \mathbf{w}, (\mathbf{p}-\mathbf{q}) \odot (\mathbf{p}-\mathbf{q}) \rangle$, where $\odot$ denotes the entrywise product. The parameters $\mathbf{w}$ control the opening width of a 2D parabola centered at $\mathbf{p}$, penalizing scores $f(\mathbf{q})$ based on the distance between $\mathbf{p}$ and $\mathbf{q}$. The greater the distance between $\mathbf{p}$ and $\mathbf{q}$, the greater the penalty on $f(\mathbf{q})$. We can interpret Equation 4.6 as a filtering operation, where $f(.)$ is a function defined on a 2D-grid and $h(\mathbf{p}-\mathbf{q})$ is a max-convolution filter. The filter kernel has a side length of $2s + 1$ (we empirically chose $s = 10$) and each filter weight at position $\mathbf{x}$ is initialized to

$$h(\mathbf{x}) = \left\langle \mathbf{w}, \left(\mathbf{x} - (s, s)^T\right) \odot \left(\mathbf{x} - (s, s)^T\right) \right\rangle, \tag{4.8}$$

a parameterized parabola rooted at the center of the kernel. Using this kernel, the distance transform at any point $\mathbf{p}$ can then be obtained by placing the filter at position $\mathbf{p}$, performing an element-wise subtraction from the function $f(.)$ and maximizing over all resulting values.

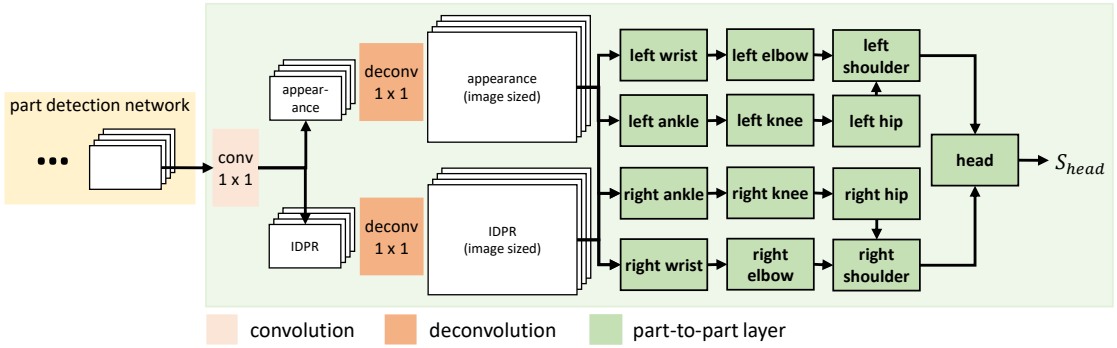



Figure 4.4: The CG-IRLS graph structure for human pose estimation. Appearance maps are summed up to form a representation for joint appearance and image dependent pairwise relation terms. After resizing all maps to the original image size, both are passed to a network of part-to-part layers (green rectangles, see also Figure 4.3) representing the structure of the human pose graph $G$. The part-to-part layers compute partial solutions for each joint.

### Full Inference

The computational graph depicted in Figure 4.3 represents just one building block of our pose estimation pipeline. The combination of multiple of these part-to-part layers - structurally arranged as a human pose graph - represents the whole pose estimation model. The resulting computational graph is depicted in Figure 4.4 and discussed in the following.

From the part detection network, we obtain a set of confidence maps representing beliefs that the part is present in different configurations of neighboring parts. From this large collection of specific belief maps, two stacks of score maps are extracted as described in Section 4.2.1, one for the appearance of the parts and one for image dependent terms. We resize all appearance and IDPR maps to the size of the original input image using a deconvolution layer that implements a bilinear deconvolution kernel. The resized confidence maps are passed into a tree-shaped network consisting of part-to-part blocks. Each green rectangle in Figure 4.4 corresponds to one part-to-part layer outputting a partial solution $S_i$ for joint $i$. Note that the structure of all part-to-part blocks represents exactly the pose graph $G$. We denote the whole pose estimation graph as CG-IRLS.

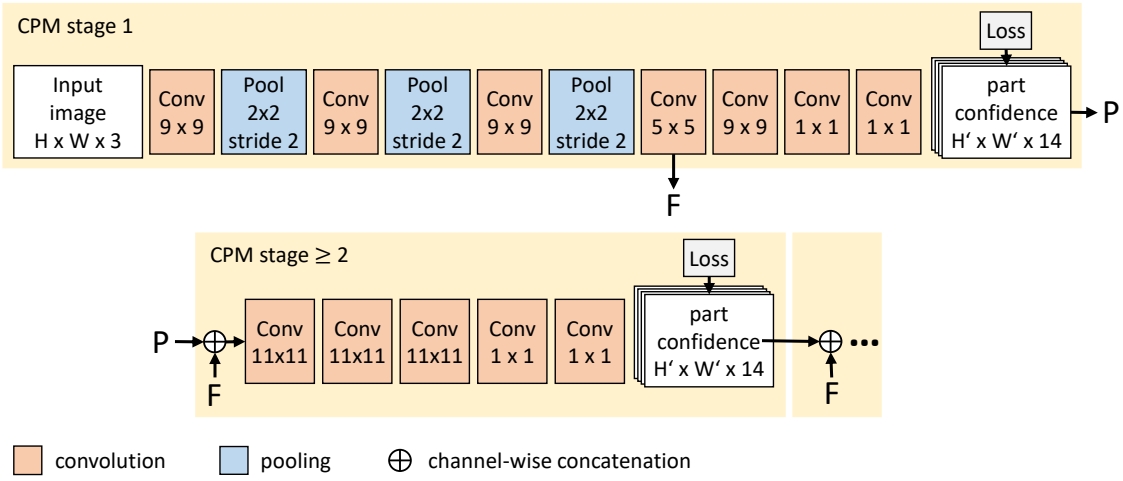



Figure 4.5: A convolutional pose machine (CPM) as proposed by [106] consists of multiple stages. The first stage (top) learns appearance features and is structurally similar to the appearance stage in Figure 4.2. It outputs confidence maps P for each joint. The following stages (bottom) concatenate the output P of their previous stage together with an intermediate feature representation F from the first stage. Multiple stages can be connected in line.

## 4.3 Convolutional Pose Machines

We discuss an alternative network structure for human pose estimation called a Convolutional Pose Machine (CPM), initially introduced by Wei et al. [106]. They propose a much simpler, yet effective network architecture to learn the structure between joints purely from data, thereby taking the final leap from a manually modeled pose estimator towards an end-to-end trainable, data-driven pose estimation approach.

The CPM network structure is depicted in Figure 4.5. It consists of several stages, usually three, and can be extended with additional ones. The first stage is designed with the same intention as the part detection stage for the CG-IDPR model and therefore shares an unsurprisingly similar architecture. The network input is a rectangular image patch depicting a person at its center. The first stage outputs a stack of joint confidence maps, one for each joint plus an additional one for the background. These confidence maps ideally contain high confidence values at positions where a joint is present in the input patch. Unfortunately, visually similar joints will usually produce high confidence values in the wrong joint maps, as will

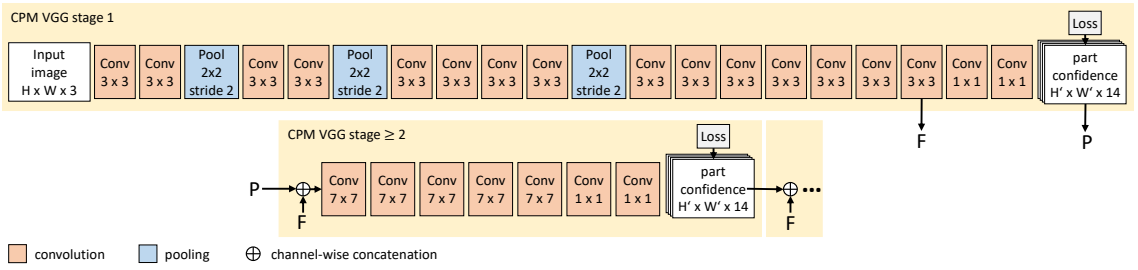



Figure 4.6: The CPM VGG network implements the CPM structure using smaller kernels where possible. As the receptive field should be kept the same as in the original formulation, using smaller kernel sizes naturally increases the network depth.

false positive predictions in general. Fixing these false detections is the job of all consecutive stages.

All stages following the first one are built the same way. The input to each stage is a collection of joint confidence maps from the previous stage, which are concatenated with intermediate appearance features from the first stage (see Figure 4.5 for details). After five convolutional layers with relatively large kernel sizes, the stage outputs the same number of joint confidence maps as the previous stage. With their large receptive field that covers almost the complete input image, the kernels in each stage learn to emphasize correct joint detections while suppressing potentially false positive scores. This mechanism improves confidence maps with every stage added to the network. Commonly, a fully CPM consists of three stages, one appearance stage, and two "structural" stages, but networks up to six stages have been evaluated by Wei et al. Adding stages leads usually to better scores, but also to an increase in convergence time during training and inference time.

The standard CPM formulation as depicted in Figure 4.5 is comprised of kernels with large sizes. This is necessary as the receptive field of the network needs to be large enough such that false detections can be suppressed based on the positions of the surrounding joints. A disadvantage is that networks with large kernels consequently also have a large number of parameters that need to be estimated, which usually involves larger amounts of training data more training time. We experimented with an alternative architecture of a CPM influenced by the findings of [93], who propose the VGG architecture by replacing large convolutional kernels with a series of smaller $3 \times 3$ kernels. Therefore, we replace convolutional kernels in stage one of the standard CPM with multiple $3x3$ kernels (see Figure 4.6), consequently

making the network a bit deeper to retain the size of the receptive field. In all successive structural stages, all $11 \times 11$ layers were replaced by multiple $7 \times 7$ layers with the same final receptive field as in the original CPM. We found that altering the receptive field or using kernels with a smaller size always yields subpar pose estimation results. We denote this alternative network *CPM-VGG* and discuss the results of this modification in the experimental section.

## 4.4 Key-pose Occurrences from Human Pose Estimates

We explore two methods for identifying key-poses in sequences of athlete pose estimates in the following.

### 4.4.1 Pose Feature Series

The first procedure is almost trivial and straight forward to implement. For each key-pose, we translate the associated pose feature into a time-series of key-pose confidence values with the property that the confidence has to be maximal for frames that depict the key-pose and lower otherwise. Take for instance key-pose one depicted in Figure 3.5. This key-pose was defined by an expert based on the pose feature "270-degree angle of the upper arms". This specific angle is directly deducible from the joint estimates for shoulder and elbow. Given a sequence of pose estimates, we use the time-series notation $(x_t)_{t \in \mathbf{T}} = (x_1, ..., x_T)$ to define a sequence of confidence values for the upper arm angle as

$$Y = (-\sin(\alpha^{(t)}))_{t \in \mathcal{F}} \tag{4.9}$$

for each upper arm angle $\alpha^{(t)}$ (in radians) for frames $t$ in the ordered sequence of frame-indices $\mathcal{F}$. $Y$ is locally maximal if the key-pose occurs and produces lower confidence values for all other arm angles.

For all other key-poses, we can design functions that are locally maximal if a key-pose is present and produce smaller confidences otherwise. The temporal occurrence for any such design function $Y[t]$ is then simply given by the set $K$ of all local maxima $\kappa$ of $Y[t]$:

$$K = \{\kappa : Y'[\kappa] = 0 \wedge Y''[\kappa] > 0\}. \tag{4.10}$$

In order to account for double detections within temporally close frames, we apply non-maximum suppression by keeping local peaks with the largest confidence values and deleting peaks within a temporal window of $\pm 6$ frames around them.

This approach is trivial as long as a pose feature is directly observable in the pose estimate and as long as we can find a design function with the aforementioned properties[3]. A drawback of this intuitive method is that key-poses and respective pose features have to be established at the time of implementation. We use the abbreviation N-TS to denote the method of using naively designed time series from pose features.

### 4.4.2 Random Forest Classification

We propose a second method that treats key-pose identification as a random forest classification problem. The classifier learns to predict an is-key-pose label just from attributes derived from the pose estimates alone. Therefore, we build a training set from sequences of pose estimates where each frame is flagged as a key-frame or as a non-key-frame by an expert. We first normalize the location of each estimated pose hypothesis by centering it at the x-coordinate of the head and the y-coordinate of the water surface. A pose representation vector[4] $\mathbf{f}^{(t)}$ is compiled from each pose estimate by concatenating all joint locations. We additionally add the sines and cosines of the edge orientations (e.g., upper arm angles, forearm angles, et cetera) and angles between neighboring edges (e.g., the angle between the thigh and lower leg, and so on) to the feature representation. Limb orientations and angles cover the set of possible pose features defined in Section 2.2.2. The inclusion of these attributes is necessary because random forests do not have the ability to learn dependencies between different attributes in the feature vector.

Naturally, the number of occurrences of a key-pose is limited to one frame per complete swimming cycle, while all other frames in the cycle do not depict a key-pose. To counteract this rather imbalanced set of positive versus negative examples, we increase the number of positive training examples by labeling the poses in the $\pm 2$ adjacent frames of a key-pose frame also as key-poses. At the same time, we decrease the number of non-key-poses by randomly sampling from the set of all

[3]In case we want to find key-poses based on latent pose features, the method described in Chapter 3 is more appropriate.

[4]Not to be confused with the pose feature defining a key-pose.

non-key-pose examples such that we are left with approximately five times more negative examples than positives. We finally train a random forest classifier $RF(.)$ to distinguish between key-poses ($RF(.) = 1$) and non-key-poses ($RF(.) = 0$).

At inference, we build a feature representation for each pose in a video and aggregate the predicted class labels. The resulting confidence sequence is given by

$$Y = (RF(\mathbf{f}^{(t)}))_{t \in \mathcal{F}} \tag{4.11}$$

It ideally contains short bursts of consecutive positive labels. We use a low-pass filter to emphasize the positive label of the key-pose in the middle of a burst. We identify a key-pose through local maxima search and apply non-maximum suppression around high scoring poses. The abbreviation RF-TS is used to denote the method of using a random forest classifier to produce a key-pose time series.

## 4.5 Experiments

We evaluate the CG-IDRP model and compare it against CPM and CPM-VGG. Articulated human pose estimation allows for evaluating the joint estimation performance directly using established pose estimation criteria. The key-pose predictors are evaluated using the same protocol as in the last chapter.

### 4.5.1 CG-IDPR Training

Training the CG-IDPR model involves training the model components separately, for which we use 1200 images of freestyle swimmers from the *mmc-swim-channel* dataset. From this set, 900 images are used for training the network, 100 for validation, and 200 serve as a test set for evaluating the pose estimator.

Each offset between two joints is quantized using k-means clustering with eleven clusters for all arm joints and six clusters for all other joints. We crop patches from overall 13 joints (head, shoulders, hips, knees, ankles, elbows, wrists) with a patch size of $100 \times 100$ pixels and augmenting this initial set by adding variations of training patches. Therefore, we randomly rescale each training image by a factor of $[0.8, 1.2]$ and add random rotations within $+/-15$ degrees. We randomly crop negative patches from images showing empty swimming channels with different water flow velocities. Our final training patch collection covers 44.500 examples, which are

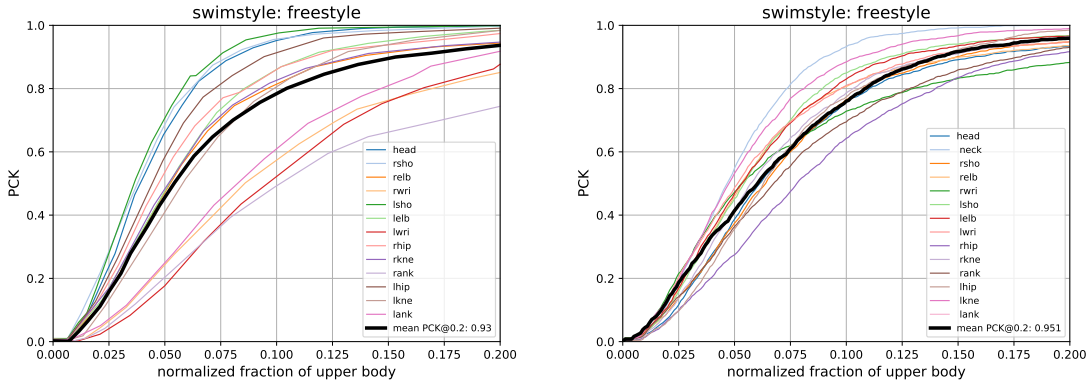



Figure 4.7: PCK curves of CG-IDPR (left) and CPM (right) networks fine-tuned on freestyle swimmers. While both networks work well in this application scenario, CG-IDPR has more difficulties detecting extremities of the athlete.

resized to the net input size of $256 \times 256$ and tagged with a label from the label set $\mathcal{L}$ The network was implemented using Caffe [61] and is trained for 40.000 iterations, batch size 16 and an initial learning rate of 0.001, which is reduced by a factor of 10 every 10.000 iterations. After the network has converged, the parameters are fixed and the last two layers are remodeled to fully convolutional layers.

The second training step entails deriving deformation terms and other weight parameters. Therefore, we process the training images with the appearance network and generate joint appearance and IDPR maps. The deformation parameters and weights are learned as described in Section 4.2.1. The weight parameters of the deformable part model are optimized on the training images using three bootstrapping rounds.

The CPM model used for comparison in this paper was also fully implemented and trained using the Caffe Framework [61], while CPM-VGG was implemented using the Tensorflow framework [28]. Both networks were initially trained on the LSP dataset and fine-tuned on the 900 freestyle swimmers in the training set. As a consequence of the LSP pre-training, they predict an additional joint position of the neck.

### 4.5.2 Joint Localization

We evaluate the joint localization performance of CG-IDPR and the classical CPM in Figure 4.7 first. Therefore, patches depicting the athlete in the test set are cropped

| | CG-IDPR | CPM | CPM-VGG |
|---|---|---|---|
| head | 99.6 | 93.6 | 93.2 |
| neck | - | 100.0 | 99.9 |
| right soulder | 99.3 | 94.5 | 94.3 |
| right elbow | 94.0 | 93.5 | 93.0 |
| right wrist | 85.5 | 88.4 | 88.1 |
| left shoulder | 100.0 | 96.8 | 96.7 |
| left elbow | 98.9 | 96.5 | 96.0 |
| left wrist | 87.8 | 94.9 | 94.7 |
| right hip | 97.3 | 91.7 | 91.6 |
| right knee | 94.8 | 95.6 | 95.1 |
| right ankle | 74.7 | 93.1 | 92.8 |
| left hip | 99.2 | 98.6 | 98.1 |
| left knee | 98.1 | 98.9 | 98.3 |
| left ankle | 91.1 | 94.8 | 94.7 |
| mean PCK | 93.8 | **95.2** | 94.7 |

Table 4.1: Comparison of PCK scores (in [%]) for CG-IDPR, CPM and CPM-VGG. All joints of 200 freestyle swimmers from the dataset are evaluated. The CPM outperforms the other networks with an average PCK score of 95.2%.

from the images and used directly as an input to the networks. The patches for the CG-IDPR network are resized by a factor of 2.5, reflecting the upscaling factor during training. The patches for the CPM model are rescaled to fit the network input of $368 \times 368$.

### Percentage of Correct Keypoints

A common metric to judge the quality of human pose estimates is the *Percentage of Correct Keypoints* (PCK, [86]) measure, sometimes called the *Percentage of Detected Joints (PDJ)* metric. This metric normalized all joint residual magnitudes with the upper body length of a person, defined by the distance between right shoulder and left hip. The normalized residual threshold $\delta$ is then compared on the x-axis against the percentage of correctly detected joints for each threshold. Similar to the PCKF, the PCK is usually evaluated for different thresholds of the upper body size. The most common threshold is $\delta = 0.2$ (PCK@0.2), which is considered by most authors for comparing the performance of different pose estimators.

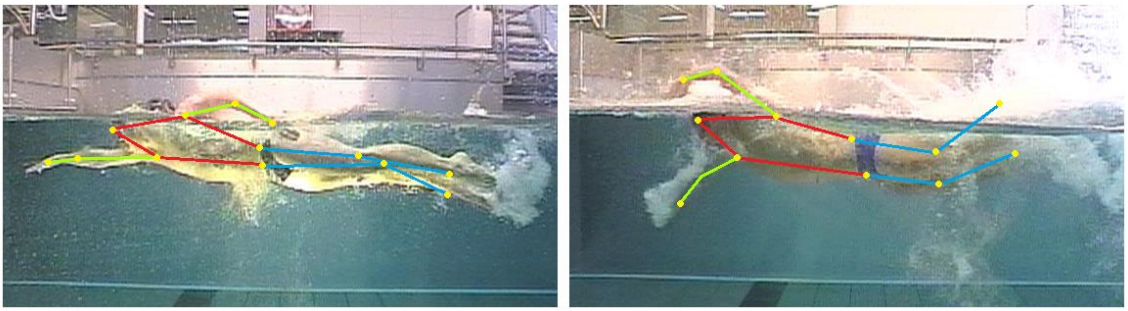


Figure 4.8: Two swimmer poses estimated by the CG-IDPR model.

#### PCK Evaluation

Figure 4.7 (left) shows that we achieve a PCK@0.2 of 93.8% with CG-IDPR on our test set of 200 freestyle swimmer images. For the same threshold, the authors who introduced IDPR terms originally [15] report a PCK value only for elbows and wrists on the LSP dataset [62], which in their implementation equals 94.9% and 92.0%, respectively. Our implementation achieves average scores of 96.5% and 86.6% for both joint types, respectively. Detecting the wrists is much more challenging for swimmers, especially as the right wrist is often occluded or in general difficult to detect.

The CPM (Figure 4.7, right) outperforms the CG-IDPR model by 1.4% PCK. Note that this model includes detections for the neck of the athletes, which was added manually to the ground truth data. A notable difference in both models can be found in the extremities, where the CG-IDPR model produces inferior prediction results. Comparing joint scores for both models in Table 4.1, we find the CPM performs sub par especially for the head (-7%) and right hip joint (-5.6%). We additionally evaluated the PCK for the CPM-VGG model for comparison. We find the scores to be comparable with the standard CMP models, coming out just a bit worse (-0.5%) on average. Two qualitative examples of the proposed CG-IDPR model are depicted in Figure 4.8.

### 4.5.3 Key-Pose Localization

For the task of key-pose identification, we use the same 30 freestyle swimmer videos from a subset of the *mmc-swim-channel* dataset as in the previous chapter. We

| | key-pose | | | | |
|---|---|---|---|---|---|
| | # 1 | # 2 | # 3 | # 4 | mean PCKF |
| ML-AL PCKF@0.02 | 0.760 | 0.622 | 0.598 | 0.520 | 0.625 |
| N-TS PCKF@0.02 | 0.801 | 0.813 | 0.779 | 0.615 | 0.752 |
| RF-TS PCKF@0.02 | **0.810** | **0.827** | **0.793** | **0.631** | **0.765** |
| ML-AL PCKF@0.03 | 0.898 | 0.717 | 0.727 | 0.662 | 0.751 |
| N-TS PCKF@0.03 | 0.913 | 0.938 | 0.910 | 0.890 | 0.912 |
| RF-TS PCKF@0.03 | **0.920** | **0.941** | **0.921** | **0.898** | **0.920** |

Table 4.2: PCKFs for N-TS, RF-TS and the best automatic key-pose estimator ML-AL (no prior) from the last chapter. On average, RF-TS outperforms both other approaches. The last column reports the average precision over all key-poses for an allowed inlier threshold of ±10 frames around a ground truth key-pose annotation.

evaluate different methods of key-pose extraction based on pose estimates of the CG-IDPR network.

For the N-TS method, we design different time series functions for all key-poses of freestyle swimmers as depicted in Figure 3.5. A time series is generated for each key-pose in each video and smoothed with a Gaussian kernel with standard deviation $\sigma = 1$. The key-frames are identified by local maximums in the time series and compared to ground truth labels using the PCKF measure. For the random forest key-pose classifier RF-TS, we use a leave-one-out cross-validation scheme, where the classifiers are trained on 29 videos and evaluated on the remaining test video. For each key-pose type, we use a random forest with 10 trees, maximum tree depth of two and information gain split criterion. A class decision is inferred by a majority vote between all trees, weighted by the fraction of all samples in the decision leaf of each tree. Similar to N-TS, the resulting time series are smoothed using a Gaussian kernel with a standard deviation of $\sigma = 1$. The final PCKF scores are averaged over all validation splits. Both methods are compared to the best fully automatic key-pose predictor ML-AL from the last chapter in Table 4.2.

### PCKF Comparisons

On average, both proposed schemes for key-pose estimation outperform ML-AL by a wide margin. The best result is obtained using the RF-TS classification approach, which outperforms ML-AL by 16.9% on average. RF-TS is also better than N-TS, although the average margin is much smaller (+0.8%). We found that the reason

for this lead is that the random forest adapts easier to the detection bias of a pose estimator. For example, if a human tags a 90-degree angle of the upper arm as the pose feature of a key-pose, a pose estimation system might on average detect these key-poses with a slightly different angle. A learning system is able to adapt to these differences while the naive approach N-TS just places a key-pose occurrence at a hard 90-degree measurement.

Last but not least, we compare the precision of all three approaches. Again, we use an inlier threshold of $\pm 10$ frames, i.e., a key-pose is counted as a true positive if is lies within $\pm 10$ frames of a ground truth annotation and no other estimate is closer to that annotation. Otherwise, a key-pose estimate is counted as a false positive. N-TS and RF-TS achieve a precision of 0.92 and 0.9, respectively. Both procedures introduce more false positive key-pose estimates than ML-AL, which achieves a precision of 0.985. As the determination of a key-pose is directly based on the pose estimate, every pose estimation error can introduce false key-poses.

## 4.6 Summary

In this chapter, we discussed how key-poses can be directly derived from a prediction of the human pose itself. We introduced a network implementation of a classical deformable part model with additional pairwise relations terms for predicting joint neighborhoods. We briefly discussed convolutional pose machines, a fully convolutional, end-to-end trainable alternative pose estimator. In our experiments, both networks perform well for the task of pose estimation on freestyle swimmers and produce similar quantitative results, with the CPM performing slightly better.

The determination of key-poses directly from pose estimates is tested in two ways. Firstly, the pose feature is queried directly, translated into a time-series and key-poses are determined by local maxima search. As an alternative, we examined a random forest classifier to predict a key-pose without knowledge about the underlying pose feature. Therefore, the poses in video footage are transformed into feature representations and a random forest is trained to automatically distinguish between key-poses and non-key-poses. We can experimentally show that both approaches perform similarly well, with the classifier beating the intuitive method. Both outperform the poselet detector presented in the last chapter by far.

# 5 — Kinematic Pose Rectification

A pose estimation system enables us to continuously estimate the human pose in a video. While state-of-the-art pose estimation algorithms often perform well on benchmark video footage and less complex scenes, the output of such algorithms can be inaccurate and partially incorrect in a visually demanding training scenario. Human pose estimation in aquatic environments is made more difficult by noise from air bubbles, water splashes, constant self-occlusion and refraction, which often impede the estimation of the human joints and consequently the retrieval of key-poses. The unique perspective on the swimming channel entails an additional challenge. The side view exhibits visually ambiguous poses for antisymmetrical body movement, thereby inducing many false detections.

In this chapter, we analyze different error modes of the CPM pose estimation system that are common in swimming channel footage[1]. A pipeline of optimization problems is proposed to rectify different types of estimation errors. The experimental section reports PCK improvements for all swim styles and attests that key-pose prediction in general benefits from pose rectification.

## 5.1 Related Work

Deep architectures have been used for improving pose estimation in videos. For instance, Song et al. [94] stack multiple contiguous frames as parallel input for a network, while [9] leverage a recurrent network structure for improving pose estimates. Iqbal et al. [58] incorporate human action into a deformable part model to improve pose estimation in videos and propose a procedure for jointly training both

[1]Parts of the analysis and algorithms discussed in this chapter have been published in [113, 114].

pose and action in one model. Gkioxari et al. [43] feed additional pose priors from previous detections together with an image into a CNN. While all these methods attempt to enforce consistency by leveraging some type of smoothing constraints, others like Pfister et al. [80] favor optical flow as an additional input modality. Recently, multi-person tracking has gained a lot of attention from the scientific community. Iqbal et al. [59] use LP optimization on a spatiotemporal pose graph for tracking multiple persons. Girdhar et al. [41] use Mask R-CNN[51] for keypoint prediction in small clips and link them over an entire video using a lightweight tracking scheme. While both multi-person tracking solutions are structurally similar to our method, none of them allows for explicit label swapping.

Pose rectification can also be seen as a "data cleaning" task. To this, recurrent frameworks [73], conditional Boltzmann machines [96], Kalman filtering [6], dimensionality reduction [3] or data-dependent random forests [29] have been successfully applied to identify and rectify outliers in human poses and other modalities. Specifically in sports, Hwang at al. [56] combined global and local pose estimation to refine joint predictions of athletes. Human pose estimation in live sports footage has been addressed by Fastovets et al. [31], who propose a generative learning algorithm for athlete tracking. In earlier work [113], we performed pose rectification for athlete by means of a convex optimization problem based on motion kinematics of an athlete. This solution involves a strictly causal dynamic program, which does not allow for incorporating detections from future frames.

## 5.2 Pose Estimation Error Taxonomy

We embark this chapter by discussing different types of pose prediction errors that are common with novel pose estimation systems like Mask R-CNN [51] or CPM [106]. The idea of this section is to provide the reader with a better sense of error types and frequency that may occur in sports videos and to convey a deeper understanding of why it is necessary to address them.

We focus our error assessment on only two of the four major swimming styles because we found that the pose detectors produced most errors on them, quantitatively and qualitatively. However, the error classes we observe for the other swimming styles are at least a subset of what is discussed in the following. We performed an analysis of joint errors on pose estimates produced by a CPM, which we

fine-tuned on a set of 56 image sequences covering overall 4500 images of freestyle and backstroke swimmers in the *mmc-swim-channel* dataset. As a dataset containing 4500 images is relatively small, it is desirable to use all of them for error investigation. To abide best practices, one pose estimate for each image can be obtained by three-fold cross-validation. Therefore, the dataset is partitioned into three complementary subsets. Either two subsets are joined to a training set which is used to train and validate a pose estimation model. Pose predictions can then be obtained for the third remaining set, denoted the test set. For one split, a CPM initially pre-trained on the Leeds Sports Pose dataset [62] is fine-tuned using the training set. Approximately 10% of the training set was chosen as a validation set, usually covering one to two full swimming sequences.

In Figure 5.1, different types of errors are depicted. They usually fall into one of four categories:

- A **joint transposition** occurs if the estimator swaps joints that are visually similar. For instance, the right wrist is falsely detected as the left wrist and vice versa. In our pose parametrization, there exist six visually similar partner joints, which are the wrists, elbows, shoulders, right and left hip, knees, and ankles. Swaps are often correlated with visually challenging poses, where the detector is not able to properly distinguish between body sides. This error class is, therefore, most prevalent in antisymmetrical motion, e.g., freestyle or backstroke swimming. An error is classified as joint transposition if and only if both joint predictions lie within the true positive error range of the partner joint, respectively. Swaps may occur for one or multiple partner joints at the same time. If correctly identified, they can be fixed by just swapping the joint labels for the respective joints.

- A joint estimate is considered an **outlier**, if its distance to the ground truth joint location is too large. This includes cases where two partner joints are placed at the ground truth position of one of both joints. Outliers are classical false positive detections which can often be corrected using robust interpolation. This error typically occurs as a standalone event as well as in bursts, where the pose estimator completely losses a joint over multiple frames.

- A ground truth joint location is a denoted a **false negative** or **miss** if there exists no detection with the correct label in the proximity of the ground truth.

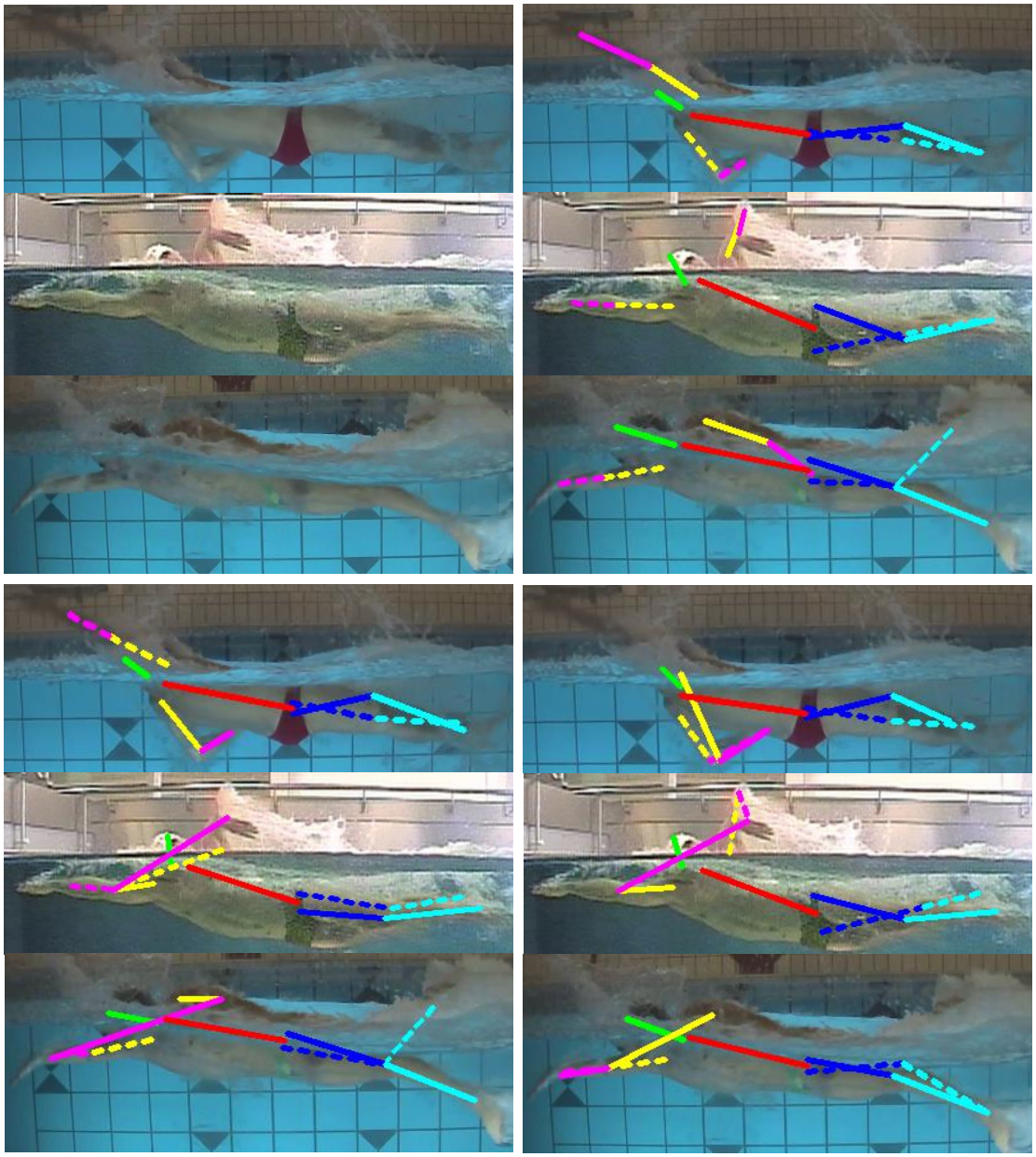


Figure 5.1: Qualitative examples of different errors. Top left: three original images. Top right: ground truth annotations. Bottom left: pose prediction by fine-tuned CPM [106]. Bottom right: pose prediction by fine-tuned Mask R-CNN [51]. The results from both pose estimation networks both illustrate a variety of estimation errors.

False negatives are usually the counterpart to outliers and can be equally corrected by using interpolation.

- All other joint detections that are not predicted precisely a the ground truth location but in close proximity fall in the last category. This **location jitter** has a joint-specific variance and usually leads to volatile joint trajectories. While some joint estimation jitter may be acceptable, a reasonable requirement for a good joint detector is that a set of joint predictions should on average have a zero mean deviation from the respective ground truth annotations. As can be seen in the following, this requirement is easily violated in visually challenging scenarios.

Joint transposition and outlier errors are typically considered false positives, while misses are false negative detections. In the context of the PCK measure, the rectification of these error categories leads to a better recall at the largest inlier threshold. Improving joints on a finer scale by removing location noise from the trajectory mainly raises the PCK curve for smaller thresholds.

### 5.2.1 Frequency

One important question to ask is how common different error types are. If each error class only rarely occurs, then correcting them would be a trivial task. Figure 5.2 visualizes the x and y coordinates (blue and orange) of an elbow and a wrist of a freestyle and backstroke swimmer, respectively. The lines in the left column of the plots depict the raw output of the CPM, while the transparent corridors indicate the true joint trajectories. The width of each corridor represents a deviation if $\pm$ 0.2 of the upper body length around the ground truth. These graphs are very exemplary for the error frequency we observe in the data: While joint swaps and outliers can appear as standalone events, more often than not they appear in bursts over a couple of frames.

In addition to the number of errors, the mixture of different error types proves challenging to tracking algorithms. As becomes apparent in Figure 5.2, joint transposition errors mix with single and consecutive outliers as well as general detection noise. This mixture of errors makes it especially difficult to quantify the frequency of each error type. From the PCK curves in the experimental section, we know that

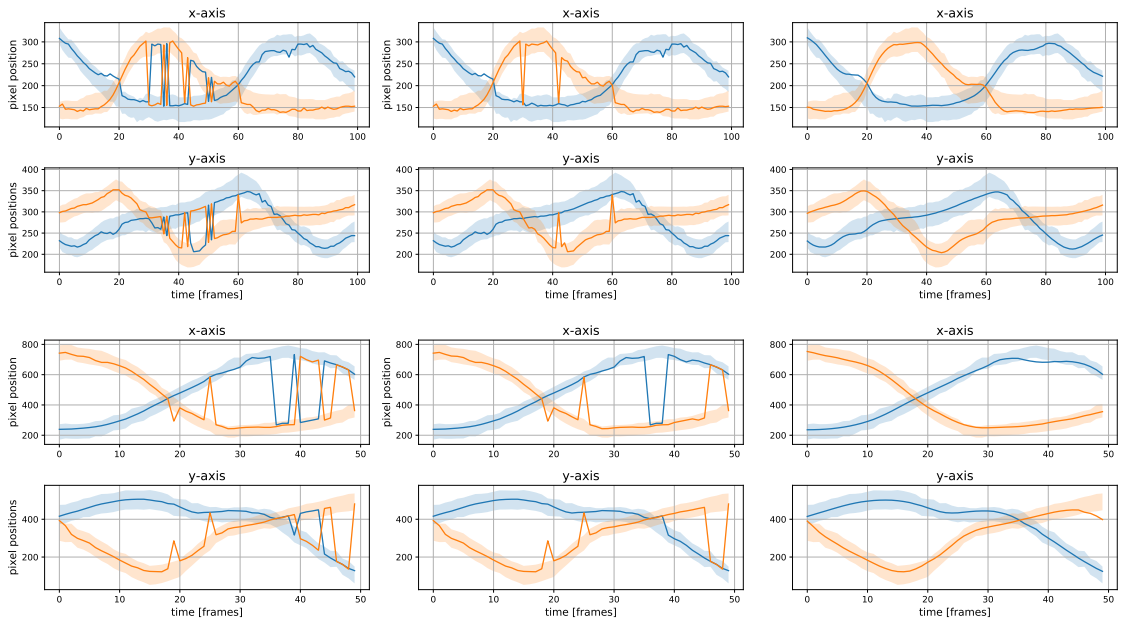



Figure 5.2: A sequence of joint coordinates of a freestyle swimmer's left (blue) and right (orange) elbow (top two rows) and a backstroke swimmer's wrists (bottom two rows). Row 1 and 3 depict x coordinates, row 2 and 4 the y coordinates. From left to right: Original sequence as predicted by a CPM fine-tuned on swimmer data, after swap correction, after outlier and noise correction. Colored corridors depict the ground truth timeseries with ± PCK@0.2.

about overall 4% to 5% of all detections have to fall in either the outlier or the joint swap category.

### 5.2.2 Correlation

We can obtain a different perspective on error behavior by examining the correlation between joint residual magnitudes. Figure 5.3 depicts the Pearson correlation coefficients for error magnitudes of all joints in a pose configuration. For both freestyle and backstroke swimmers, we find a correlation of up to 0.8 for the upper body joints and slightly larger correlation for the lower body. Especially between partner joints, the correlation coefficients are considerably larger than for other joint pairings. Large Pearson coefficients indicate a strong linear relationship between residuals, i.e., if there is a displacement of one joint, the partner joint is displaced with equal magnitude. While we might expect some error correlation for a well work-

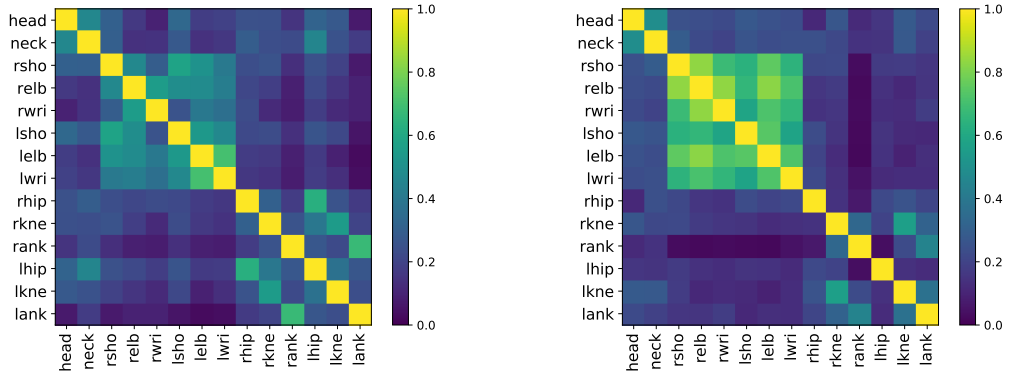



Figure 5.3: Pearson correlation coefficients for residual magnitudes of different joints for freestyle (left) and backstroke (right) swimmers. Larger correlation coefficients emerge for the upper body joints and especially for partner joints, and separately for the lower body joints.

ing pose estimator, the large coefficients between partner joints and the decoupled correlation of upper and lower body are somewhat unexpected.

### 5.2.3 Directional Offsets

We examine the general detection noise between ground truth and joint estimates. A fair assumption to make about a decently well functioning pose estimator is that over an adequately large sample of joint detections, we can expect the mean over all estimation residuals to be zero. This expectation does not hold for swimmers in a swimming channel, as there is a measurable non-zero mean offset of residuals *along* body parts (=parts are estimated foreshortened). We demonstrate this effect with a simple analysis. For a joint of type $j$, let $\tilde{\mathbf{l}}_j$ be the ground truth location and let $\mathbf{l}_j$ be the predicted location. The adjoining ground truth body part $\tilde{\mathbf{q}}_j$ can be defined as $\tilde{\mathbf{q}}_j = \tilde{\mathbf{l}}_j - \tilde{\mathbf{l}}_{\mathcal{P}_j}$, where $\mathcal{P}_j$ is the parent joint of $j$. The orientation of a body part is given by the angle $\alpha = \angle\tilde{\mathbf{q}}_j$. The residual between a predicted joint and its associated ground truth can be normalized with the orientation $\alpha$ by multiplication with a rotation matrix $R_\alpha$, yielding the residual $\mathbf{r}_j$:

$$\mathbf{r}_j = R_\alpha^T \tilde{\mathbf{q}}_j \tag{5.1}$$

| | left wrist | right wrist | left ankle | right ankle |
|---|---|---|---|---|
| mean residual x | 3.969 | 5.690 | 3.172 | 2.065 |
| mean residual y | 0.119 | -0.505 | 0.242 | -1.216 |

Table 5.1: Mean residual error (x,y) for orientation normalized wrist and ankle residuals of freestyle swimmers. An offset of up to 5.6 pixel along the body part (x-axis) can indicates that joints are estimated foreshortend. Note that all offsets are determined for size normalized swimmers with a reference upper body size of 100 pixels. Hence, a deviation of one pixel in this table is equivalent to a real error of 1% of the swimmer's actual upper body size.

We perform this rotation for each residual offset in the dataset and normalize each residuals location with the ground truth position of the joint:

$$\mathbf{r}_{j,norm} = \mathbf{r}_j - \tilde{\mathbf{l}}_j \tag{5.2}$$

Rotating all residual offsets such that the orientation of the adjoining ground truth parts equals zero and by aligning all residuals at the ground truth location allows for determining a mean residual offset, which is summarized in Table 5.1 for the extremities of freestyle swimmers. Note that all offsets are determined for size normalized swimmers with a reference upper body size of 100 pixels. Hence, a deviation if one pixel in Table 5.1 is equivalent to a real error of 1% of the swimmer's actual upper body size.

## 5.3 Joint Refinement

In this section, we will discuss a pipeline of optimization problems, each of which is designed for rectifying the aforementioned error types. Therefore, a CPM is used to produce joint estimates for each frame in a video depicting an athlete (Figure 5.4, top). All optimization problems are merely defined on the raw 2d joint location output of a pose estimator.

### 5.3.1 Swap Optimization

The following optimization problem addresses the problem of joint transposition. The idea behind the proposed method is to first construct a graph where vertices represent joint detections and edges connect joint detections of partner joints over

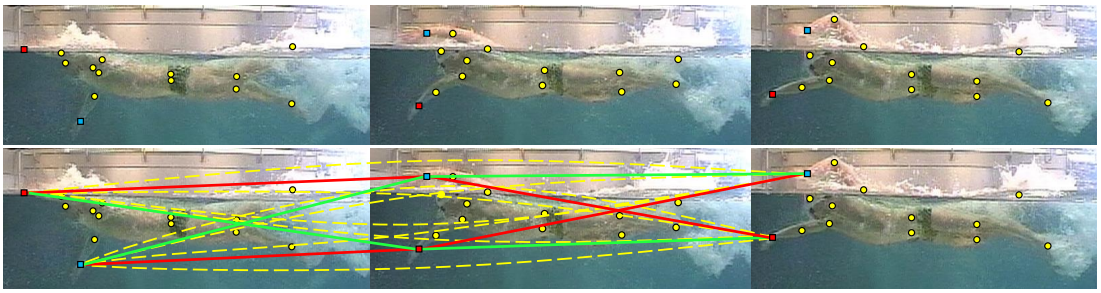


Figure 5.4: An example for a kinematic partner graph for wrist joints. Top: Three consecutive images with joint predictions. Wrists marker colors indicate joint labels as detected by [106] (swap error in left image). Bottom: ILP edges for wrists. Green edges connect joint predictions from adjoining frames. Red edges explicitly model label swaps between adjoining frames. Transitive edges (yellow) skip one frame, connecting joints from frames $t$ with frames $t + 2$.

multiple frames. The edges are weighted with probabilities that reflect whether joints belong to the same trajectory. We use an optimization procedure to delete edges from the graph, thereby partitioning it into the most probable joint trajectories for each partner joint.

### Kinematic Partner Graphs

We define a *kinematic partner graph* as follows. The denomination refers to the fact that edges in this graph model the relationship between partner joints and that the weights for each edge are derived from the motion kinematics of the athlete. A CPM is used to obtain sets of joint detections $D_t$ for each frame $t$ in a video. Each detection $d_j^{(t)} \in D_t$ of joint type $j$ is associated with a joint pixel location $l_j^{(t)} = (x_j, y_j)^T$ in the image. The function $j' = \nu(j)$ returns the partner joint $j'$ of joint $j$. For instance, if $j$ is the identifier for the left wrist, $\nu(j)$ returns the identifier for the right wrist. A kinematic partner graph is defined by edges connecting joint detections of the same type and partner joint detections over adjoining frames.

We differentiate between two edge types. Let a set of *velocity edges* $E_v$ connect detections over consecutive frames:

$$E_v = \{(d_j^{(t)}, d_{j'}^{(t+1)}) : d_j^{(t)} \in D_t \wedge d_{j'}^{(t+1)} \in D_{t+1} \wedge j' \in \{j, \nu(j)\}\} \tag{5.3}$$

The denomination of a velocity edge stems from the fact that we will use the first derivative of joint locations as weights assigned to all edges in $E_v$. Edges in $E_v$ represent the trajectory of all joint detections as originally predicted by the pose estimator. Additionally, $E_v$ includes connections between a detection $d_j^{(t)}$ of joint $j$ in frame $t$ and its partner joint detection $d_{\nu(j)}^{(t+1)}$ in frame $t+1$, which models the possibility of joint transpositions. Hence, there are four velocity edges for each pair of partner joints in two consecutive frames.

We add additional transitive edges called *acceleration edges* $E_a$ to the partner graph. Similar to velocity edges, the denomination was choosen to reflect that the edge weights for acceleration edges are derived from second order derivatives of joint locations. We define the set $E_a$ using an unusual edge notation as follows:

$$E_a = \{(d_j^{(t)}, d_{j'}^{(t+2)})_{j''} : d_j^{(t)} \in D_t \wedge d_{j'}^{(t+2)} \in D_{t+2} \wedge j', j'' \in \{j, \nu(j)\}\}. \tag{5.4}$$

Acceleration edges connect detections of joint $j$ in frame $t$ with joint detections of the same joint type as well as the partner in frame $t+2$. Note that we deliberately added a subscript $(.,.)_{j''}$ to the edge definition. It indicates that there are two edges connecting $(d_j^{(t)}, d_{j'}^{(t+2)})$, namely $(d_j^{(t)}, d_{j'}^{(t+2)})_j$ and $(d_j^{(t)}, d_{j'}^{(t+2)})_{\nu(j)}$. Both these edges connect $d_j^{(t)}$ with $d_{j'}^{(t+2)}$, but have different weights. The weights are determined by which of the two joint detections $d_j^{(t+1)}$ and $d_{\nu(j)}^{(t+1)}$ at frame $t+1$ better fit the respective trajectory. This peculiarity leads to overall eight acceleration edges connecting each pair of partner joints over two frames. A fully build partner graph for the wrists in three consecutive images is depicted in Figure 5.4, where red and green lines are velocity edges and yellow lines depict acceleration edges.

### Edge Weighting

The denomination of velocity and acceleration edges in the previous section is not accidental, as we use first and second order numerical derivatives of the joint locations to determine edge weights as follows.

**Pose preprocessing.** All predicted joint locations are resized relative to a reference upper body size $s_{ref}$ to account for athletes of different size in images. In our experimental setup, the size of an athlete in a video is approximately constant. Hence, let $s_{up}$ be the median length of an athlete's upper body size in a video, determined by the distance between right shoulder and left hip. Then, each original

joint estimate at location $\hat{\boldsymbol{l}}_j^{(t)}$ for all joints $j$ in all frames $t$ is resized to

$$\boldsymbol{l}_j^{(t)} = \frac{s_{ref}}{s_{up}} \hat{\boldsymbol{l}}_j^{(t)}. \tag{5.5}$$

The estimation of $s_{up}$ may be more complex if athletes are filmed in different training scenarios. For example, if pan shots are used and the size of the athlete in the footage changes over time, $s_{up}$ needs to be continuously estimated, for instance by a smoothing spline over predicted upper body lengths.

**Kernel density edge weights.** A weight for each edge can be computed from first and second order numerical derivatives of joint locations. Let the velocity $v_j^{(t)}$ of a joint $j$ in frame $t$ be approximated by

$$\boldsymbol{v}_j^{(t)} = \nabla \boldsymbol{l}_j^{(t)} = \boldsymbol{l}_j^{(t)} - \boldsymbol{l}_j^{(t-1)} \tag{5.6}$$

and the acceleration $a_j^{(t)}$ be the rate of change in velocity, thus

$$\boldsymbol{a}_j^{(t)} = \nabla^2 \boldsymbol{l}_j^{(t)} = \boldsymbol{v}_j^{(t)} - \boldsymbol{v}_j^{(t-1)} = \boldsymbol{l}_j^{(t)} - 2\boldsymbol{l}_j^{(t-1)} + \boldsymbol{l}_j^{(t-2)}. \tag{5.7}$$

A kernel density estimator (KDE) based on the parabola shaped Epanechnikov kernel $K(v; h) = 1 - v^2/h^2$ and a bandwidth of $h = 1$ is used to determine a probability distribution for observed velocity magnitudes $|\boldsymbol{v}_k|$ (=speed) in training videos. The distribution is queried to determine probabilities for approximated speeds from both detection locations connected to each velocity edge in $E_v$. After all weights are assigned, edge normalization for all velocity edges connecting partner joints from frame $t$ with frame $t + 1$ is performed. Therefore, all edge weights from velocity edges connecting two partner joints, i.e, $\{(d_{j'}^{(t)}, d_{j''}^{(t+1)}) : j', j'' \in \{j, \nu(j)\}\}$, are normalized such that the sum of all weights equals one.

The same weighting process is repeated for all acceleration edges in $E_a$. A KDE for each joint type is queried to determine probabilities for the estimated acceleration magnitude $|\boldsymbol{a}_j|$. When using acceleration edges to connect detections in frame $t - 2$ with detections in frame $t$, Equation 5.3.1 tells us that detections from frame $t - 1$ contribute to the approximation of acceleration. Hence, two transitive edges with distinct weights are generated connecting any transitive joint pairing. An example is shown in Figure 5.5. Both blue edges $f_1$ and $f_2$ connect the same two detections but have different edge weights, depending on the detections at frame $t + 1$ that

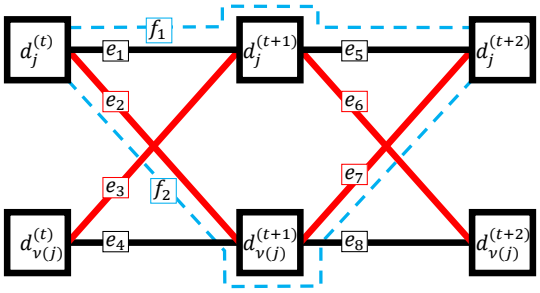



Figure 5.5: Velocity edges $\{e_1, ..., e_8\}$ between joint detections $d$ of partner joints $j$ and $\nu(j)$ in three consecutive frames $(t, t+1, t+2)$. Colors indicate predicted trajectories (black) and swap-edges (red). Transitive acceleration edges $f_1$ and $f_2$ (blue) connect $d_j^{(t)}$ with $d_j^{(t+2)}$. For clarity, only two out of eight transitive edges are shown.

contribute to the acceleration weight. We will use constraints to guarantee that only one transitive edge will be included in the final joint trajectory.

### Edge Constraints

The weighted partner graph defined in Section 5.3.1 is constructed from velocity and acceleration edges connecting partner joints over all configurations in a sequence. Recall the original idea that an optimization problem is used to delete edges from the full partner graphs. In order to guarantee that we are left with plausible and valid trajectories after optimization, we define some constraints that the optimizer has to consider.

In order to be able to intelligibly explain constraints without requiring an overly intricate notation, we will use the exemplary notation from Figure 5.5. This figure depicts an abstract sketch of two partner detections (large rectangles) in three consecutive frames (from left to right). Velocity edges $e_i$ (black and red) connect all detections from adjoining frames. Additionally, two transitive acceleration edges $f_1$ and $f_2$ are also added in this sketch. For clarity, all other transitive edges have been omitted. We will use the edge notation from this Figure to carve out optimization constraints. Note that $e_i$ and $f_i$ pose as indicator variables. If we want an edge to be existent, the corresponding variable equals 1 and 0 otherwise. In the end, the full constraint set holds all constraints for all frame triples and all partner joints in our partner graph.

The first set of constraints model mutual exclusivity between velocity edges: A detection of joint $j$ in frame $t$ can not be connected to both joint $j$ and joint $\nu(j)$ in frame $t+1$ at the same time. Using the nomenclature from Figure 5.5, this translates to:

$$\begin{aligned} e_1 + e_2 &\leq 1 \\ e3 + e4 &\leq 1 \end{aligned} \tag{5.8}$$

Both these constraints are violated if mutually exclusive edges appear together in the final solution. This set of constraints thereby guarantees that the graph is partitioned into separate joint trajectories.

The second set of constraints guarantees that the correct acceleration edges matching a selection of velocity edges are considered. If, for instance, edges $e_1$ and $e_5$ are included in a final trajectory, then the weight of transitive edge $f_1$ has to be considered in the optimization problem. This constraint is modeled as

$$e_1 + e_5 - f_1 \leq 1 \tag{5.9}$$

While this constraint models the necessary presence of transitive edges, they do not prevent the arbitrary inclusion of other transitive edges. This motivates the third and last set of constraints for mutual exclusivity of transitive edges. This constraint only allows two transitive edges per partner pairing and frame triple to be included in the final solution. It follows that only two of the eight possible transitive edges per detection triple can be included:

$$\sum_{i=1}^{8} f_i \leq 2. \tag{5.10}$$

All constraints allow for a feasible partition of the graph and thereby a trajectory for each partner joint.

### Optimization

We have defined a kinematically weighed partner graph and a set of constraints. Towards to goal of partitioning the graph into trajectories for each joint, we use an

integer linear program (ILP) as follows. We optimize over the objective function

$$\begin{aligned} &\text{maximize } \mathbf{w}^T\mathbf{x} \\ &\text{subject to } C\mathbf{x} \leq \mathbf{b} \\ &\mathbf{x} \in \{0,1\}^n \end{aligned} \tag{5.11}$$

Here, $\mathbf{x}$ is a binary vector representing all edges in the partner graph. Each entry $x_i$ indicates whether an edge is present ($x_i = 1$) or absent ($x_i = 0$). The weight vector $\mathbf{w}$ associates an edge weight $w_i$ with each edge represented in $\mathbf{x}$. In simpler words, maximizing the objective $\mathbf{w}^T\mathbf{x}$ means adding the most probable weights of the most probable trajectories of joints in the graph. The partitioning itself is achieved with the set of constraints which are encoded by matrix $C$ and vector $\mathbf{b}$. The joint detections connected by a path of edges form the optimal trajectory w.r.t. objective 5.11.

Practically, the problem is modeled using the convex optimization modeling language CVXPY [1] with the Embedded Conic Solver (ECOS, [25]). The test sequences in the experimental section do not exceed 100 frames, for which the optimization of the kinematic graph is still tractable. For evaluating longer sequences, as necessary for video processing, the sequence of poses can be split into subsequences of length 100, overlapping with three frames. The ILP is solved for each subsequence, and the edges for the first frames are fixed to the solution obtained from the preceding subsequence.

### Edge Conflation

There is some potential for optimizing the size of the problem. The edge set of the partner graph can be reduced to sets with smaller cardinality by capitalizing the fact that we only argue over partner joints. Consider both partner detections in frames $t$ and $t+1$ in Figure 5.5. If we assume that detection $d_j^{(t)}$ is connected to $d_j^{(t+1)}$ via edge $e_1$, then it automatically follows that $d_{\nu(j)}^{(t)}$ is connected to $d_{\nu(j)}^{(t+1)}$ via $e_4$. Edges $e_2$ and $e_3$ would be deleted from the graph in this scenario. Then again, if $e_2$ and $e_3$ had larger probabilities, then they would connect the detections and $e_1$ and $e_4$ would be excluded from the solution. This binary characteristic of the problem allows for merging edges in the graph. For example, edges $e_1$ and $e_4$ can be merged into a new edge with its weight being set to the sum of both single edge weights.

This edge now represents that no swap occurs from frame $t$ to frame $t+1$. The same can be done by merging edges $e_2$ and $e_3$, yielding an edge representing a possible swap. This procedure can be implemented for all velocity and acceleration edges in the graph, reducing the number of weighted edges and associated constraints by half, leading to a more compact optimization problem which can be solved faster.

### 5.3.2 Directional Offset Minimization

We discuss the problem of directional offsets along the limbs of athletes, which is very unique for swimmers. Unfortunately, removing the directional offset is a difficult problem. At inference time, the only information that can be salvaged for minimizing the directional offset are the predicted joint locations. Just moving detections a couple of pixels along the direction of the predicted body part is not adequate, as its orientation is subject to detection noise from two connected joints.

The following simple yet effective practical method was found to diminish the directional offset. In contrast to the method proposed in [113], this approach is much simpler and yields similar results. On a separate set of training videos with ground truth joint locations $\tilde{\mathbf{l}}_j$ a CPM is used to estimate joint locations $\mathbf{l}_j$. Just like in Section 5.2, the body part $\mathbf{q}$ connected to a predicted joint is given by $\mathbf{q} = \mathbf{l}_j - \mathbf{l}_{\mathcal{P}_j}$, where $\mathcal{P}_j$ is the parent joint of $j$.

We use k-means clustering on all predicted parts $\mathbf{q}$, as we found that predicted body parts with similar appearance often suffer from the same directional error bias. For all parts in a cluster $c$, the residuals $r = \tilde{\mathbf{l}}_j - \mathbf{l}_j$ between ground truth and predicted joint are rotated using rotation matrix $R_\alpha$, where an $\alpha = \angle\mathbf{q}$ denotes the orientation of $\mathbf{q}$. This operation normalizes the direction of all residuals in a cluster by the orientation of the connected predicted part. For all orientation-normalized residuals in cluster $c$, the geometric median $\tilde{\mathbf{r}}_c$ is the point that minimizes the sum over the L1-norm of residuals. We choose the geometric median over the mean, which would be much simpler to compute, as after joint swap correction we are still left with partially large outliers from false joint detections. Finding the robust geometric median is one of the oldest non-trivial problems in computational geometry and can be approximated by using the Weiszfeld algorithm [2], which iteratively converges towards this point. We compute one median for all clusters of residual offsets.

At inference, we proceed accordingly. For a predicted joint location $\mathbf{l}_j$, the connected body part $\mathbf{q}$ and its orientation $\alpha$ are calculated. We look up the cluster

| | left wrist | right wrist | left ankle | right ankle |
|---|---|---|---|---|
| mean residual x | 3.969 | 5.690 | 3.172 | 2.065 |
| mean residual x, minimized | 2.487 | 3.352 | 0.288 | 0.931 |
| mean residual y | 0.119 | -0.505 | 0.242 | -1.216 |
| mean residual y, minimized | -0.100 | 0.098 | -0.104 | -0.152 |

Table 5.2: Mean residual error (x,y) for orientation normalized wrist and ankle residuals of freestyle swimmers. An offset of up to 5.6 pixel along the body part (x-axis) can indicates that joints are estimated foreshortend. Note that all offsets are determined for size normalized swimmers with a reference upper body size of 100 pixels. Hence, a deviation of one pixel in this table is equivalent a real error of 1% of the swimmer's actual upper body size.

assignment $c$ for the part, transform the associated directional median $\tilde{\mathbf{r}}_c$ using the rotation matrix $R_\alpha$ and add it as a directional correction factor to the predicted joint location $\mathbf{l}_j$ with

$$\mathbf{l}_j^* = \mathbf{l}_j + R_\alpha \cdot \tilde{\mathbf{r}}_c. \tag{5.12}$$

Table 5.2 summarizes directional offsets after this procedure. All offsets are determined for size normalized swimmers with a reference upper body size of 100 pixels. Hence, one pixel of deviation equals a real error of 1% of the swimmer's actual upper body size.

### 5.3.3 Temporal Robust Regression

We discuss a windowed robust regression approach for finding and replacing outliers while increasing the signal to noise ratio for each joint trajectory.

The basic notion behind local regression is that a noisy function can be approximated in small local *neighborhoods* using a low-order polynomial that is robustly estimated using a *weighted regression* approach. A neighborhood or window is defined by subsequences of $m$ equally spaced data points. Robustness is achieved by weighting points. Data points that are closer to the approximation get a larger weight and thereby larger influence on model regression coefficients. Likewise, the influence of outliers should be mitigated by giving them a small weight.

Robust local regression has been extensively studied for locally weighted scatterplot smoothing (LOWESS, [18]), for which we discuss some modifications. First, the authors in [18] state that although the polynomial algorithm was designed with robustness in mind, the approximation may still suffer from large outliers. They

choose arbitrary weighting functions, which we replace by weights emerging from minimizing the L1-norm of approximation residuals, as proposed by [45]. Second, LOWESS is computationally expensive. We propose a strided window placement with stride $\lfloor m/2 \rfloor$ to obtain a decent approximation of an arbitrary function more efficiently.

### L1-norm Minimization

Approximating polynomials with least squares has the disadvantage that the largest errors in a signal dominate the approximation result. While this might not be a problem if the signal noise is normally distributed, large outliers can cause degraded fits. A potential solution to this problem is to minimize the sum of absolute residuals, i.e., the L1-norm. In the following, robust approximations are estimated on timeseries of $x$ and $y$ coordinates of consecutive joint locations. We use vector $\mathbf{c} \in \mathbb{R}^m$ to describe a series of either coordinate. The classical approach to finding a polynomial model of degree $g$ that approximates $\mathbf{c}$ is given by the relationship

$$\boldsymbol{\varepsilon} = A\mathbf{x} - \mathbf{c} \tag{5.13}$$

where $A \in \mathbb{R}^{m \times (g+1)}$ is a design matrix, $\mathbf{x} \in \mathbb{R}^{g+1}$ are the polynomial coefficients and $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ are approximation residuals. Each line in $A$ and $\mathbf{c}$ corresponds to a polynomial of degree $g$ for one data point. A robust L1-norm solution for $\mathbf{x}$ can be found by minimizing

$$\mathbf{x}^* = \arg\min_x ||\boldsymbol{\varepsilon}||_1 = \arg\min_x \sum_{i=1}^{m} |\varepsilon_i| \tag{5.14}$$

Finding a solution to this problem is inconvenient as the objective is usually non-convex. A trick can be applied to transform the least absolute deviation into a weighted least squares problem with weights $w_i$ by applying

$$|\varepsilon_i|^p = |\varepsilon_i|^{p-2}|\varepsilon_i|^2 \overset{p=1}{=} w_i^{-1}\varepsilon_i^2 \tag{5.15}$$

which can be used to rewrite (5.14) to

$$\mathbf{x}^* = \arg\min_x ||\boldsymbol{\varepsilon}||_1 = \arg\min_x \sum_{i=1}^{m} w_i^{-1}\varepsilon_i^2 = \arg\min_x ||W^{1/2}(A\mathbf{x} - \mathbf{c})||_2^2 \tag{5.16}$$

**Algorithm 1** Iteratively Weighted Least Squares for L1-norm minimization

Given: design matrix $A$ ($A_i$ denotes the i-th row of A), coordinate sequence $\mathbf{c}$, convergence threshold $\eta$
Initialize diagonal of $W^{(0)}$: $W_{ii}^{(0)} \leftarrow 1$
$\delta \leftarrow 0.0001$
**repeat**
  $\mathbf{x}^{(t+1)} \leftarrow (A^T W^{(t)} A)^{-1} A^T W^{(t)} \mathbf{c}$
  $W_{ii}^{(t+1)} \leftarrow \max\{|A_i \mathbf{x}^{(t+1)} - c_i|, \delta\}^{-1}$
**until** convergence

where $W$ is a diagonal matrix with $W_{ii} = w_i^{-1} = |\varepsilon_i|^{-1}$. This reformulation is called weighted least squares and has a closed form solution. In this special case, the weights in $W$ depend on the model parameters $\mathbf{x}$. A solution to Equation 5.16 can be obtained utilizing the iteratively reweighted least squares (IRLS) algorithm, which is summarized in Algorithm 1. Note that for a matrix $A$, we use $A_i$ to denote the i-th row of $A$. To avoid dividing by 0 when $w_i \to 0$, we can introduce a smoothing parameter $\delta > 0$ and set $w_i = \max\{|\varepsilon_i|, \delta\}$. $\delta$ is usually chosen to be very small, e.g., $\delta = 0.0001$. The algorithm converges if the sum of residuals falls below a threshold $\eta$ between two iteration loops. IRLS is known to converge fast, usually 5 iterations are sufficient to reach convergence.

### Windowed Local Regression

The original LOWESS algorithm performs regression for each subwindow around a data point in order to replace it with a robust estimate. We found that we can use a strided robust regression with equal approximation results. Hereto, for each coordinate series $\mathbf{c}$, windows of uneven size $m$ - we use $m = 21$ for all experiments - are sampled with stride $s = \lfloor m/2 \rfloor + 1$, i.e., all windows overlap at $\lfloor m/2 \rfloor$ values. For each window, a robust approximation is obtained via L1-norm minimization as discussed in the previous section. We add all approximations together considering their original position in the coordinate series. Overlapping regions are linearly weighted. Hereto, let $\mathbf{c}_1$ be the last $n = \lfloor m/2 \rfloor$ values of an approximation which overlaps with $\mathbf{c}_2$, the first $n$ values of the next approximation. Let $t \in \{1, ..., n\}$ be an index for each value in $\mathbf{c}_1$ and $\mathbf{c}_2$. Each approximated value in $\mathbf{c}_1$ at index $t$ is multiplied with weight $w_t = (n + 1 - t)/(n + 1)$ and each approximation value in $\mathbf{c}_2$ at index $t$ with $1 - w_t$.

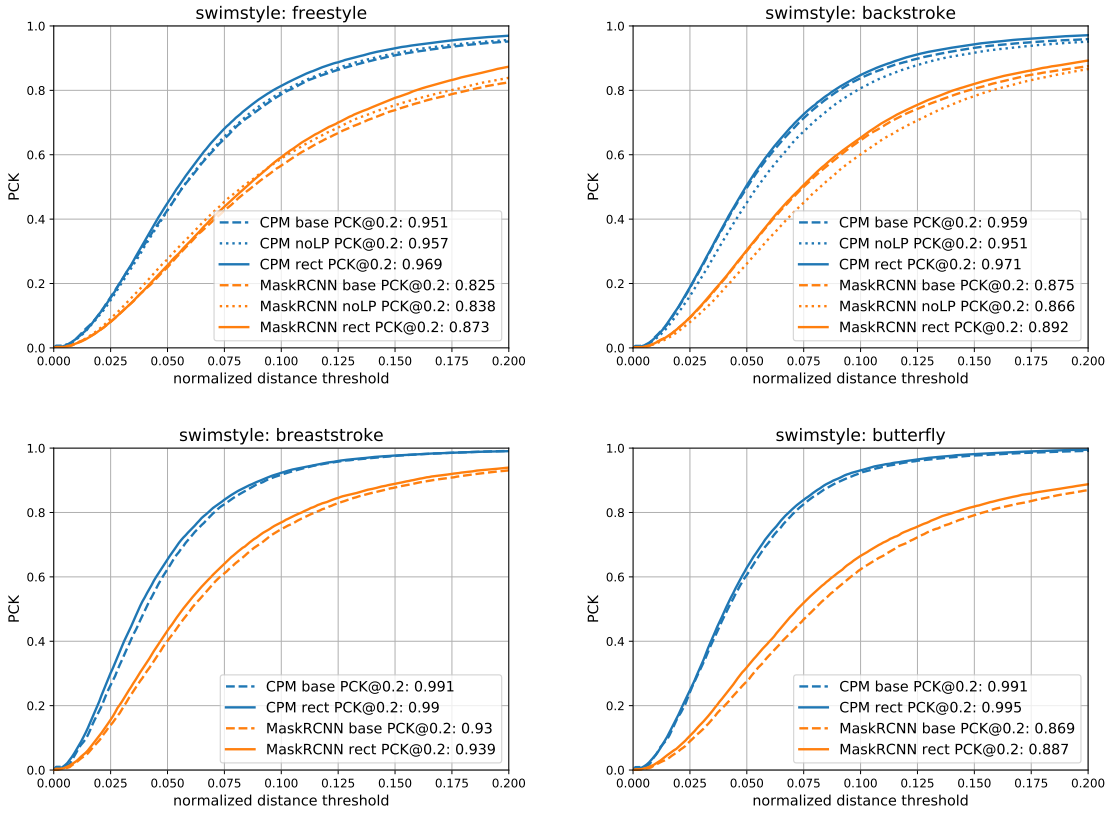



Figure 5.6: Percentage of Correct Keypoints for all four major swimming styles. Mask R-CNN (orange) is compared to CPM (blue). Dashed lines are raw pose prediction baselines, continuous lines represent the PCK after rectification. Dotted lines are results without ILP swap optimization, but with robust regression.

## 5.4 Experiments

### 5.4.1 Joint Localization

We compare two state-of-the-art pose estimation systems for joint localization: CPMs with 3 stages and additionally Mask R-CNN [51] build on a ResNet-101 [53] backbone. The CPM model is initially pre-trained on the Leeds-Sports-Pose dataset [62], the Mask R-CNN on the MSCOCO dataset. As we wish to compare the PCK of the rectification pipeline with the baselines but only have a small, fully annotated dataset of motion sequences available, we have to evaluate the pipeline on our training data. Therefore, we refine both the CPM and Mask R-CNN on each swimming style by fine-tuning it on our fully annotated sequences using a *3-fold*

*cross validation*. Hereto, the data is split into three partitions. One partition is kept as the test set while the model is refined using the remaining partitions for training and validation. Hereby, we obtain pose estimates for each frame in our fully annotated image sequences.

As the input to a CPM is an image patch depicting the complete athlete, we apply a Single Shot Detector (SSD, [72]) to all images and videos to obtain an initial location hypothesis which we then process with the CPM. This process is superfluous for Mask R-CNN, which is able to extract athlete localization proposals from the whole image within its network structure.

### PCK

The PCK on the initial estimates for both networks serves as the baseline and is depicted in Figure 5.6 (dashed). We find that there is a difference in pose estimation network performance, with CPM coming out on top of Mask R-CNN. For the CPM, symmetrical swimming styles already work very well: the PCK@0.1 is beyond 90%; almost all joints (>99.0%) are predicted within 0.2 of the upper body size. The anti-symmetrical swimming styles are more difficult to predict precisely, with a PCK@0.1 of 78.7% for freestyle and 82% for backstroke. Both hardly exceed 95% PCK@0.2. The Mask R-CNN shows surprising gains in for freestyle swimmers (+4.8% PCK@0.2), but also for breaststroke and butterfly swimming, especially compared to the marginal CPM gains. This can be attributed to the fact that the Mask R-CNN detections are much more volatile. Even between consecutive images with only very little change in content, the detections often jump unpredictably around the ground truth. This also becomes evident when comparing RMS values in Table 5.3, where the larger gain is obtained from the second rectification stage. Nevertheless, the localization gain is consistent and partially distinct for all swimming styles and both networks.

### Partner Graph Ablation

To get a better understanding of the importance of partner graph optimization, we perform a simple ablation study by applying robust joint regression from Section 5.3.1 without optimizing the partner graph. The results are depicted as dotted lines in Figure 5.6. For freestyle swimmers, the resulting PCK of 95.7%(+0.5%) for the CPM and 83.8%(+1.3%) for Mask R-CNN is slightly better than the baselines but

| | free | back | fly | breast |
|---|---|---|---|---|
| Mask R-CNN baseline | 31.84 | 34.77 | 40.70 | 26.10 |
| Mask R-CNN swap corr. | 28.91 | 32.02 | - | - |
| Mask R-CNN fully rectified | 19.15 | 20.18 | 33.31 | 16.49 |
| CPM baseline | 14.81 | 17.90 | 7.07 | 6.69 |
| CPM swap corr. | 12.08 | 11.76 | - | - |
| CPM fully rectified | **9.13** | **9.55** | **5.77** | 6.65 |
| Zecha et al. [113] | 10.74 | 11.34 | 5.78 | **6.56** |

Table 5.3: RMS values of the Euclidean distance between prediction and ground truth for all joints.

worse than the results of the full rectification pipeline. For backstroke swimmers, the PCK drops to 95.1%(-0.6%) for the CPM and 86.6%(-0.9%). We conclude that partner graph optimization is not only beneficial to improving pose estimates, it can also be necessary to avoid poor rectification results.

**RMS**

We evaluate each step in our pose rectification pipeline separately and compare the results against [113]. The joint transposition ILP is applied to freestyle and backstroke swimmers, directional offset minimization and spatiotemporal regression are applied to all swimming styles. Table 5.3 compares the root mean square values of Euclidean distances between prediction and ground truth. Note that the RMS was measured for size normalized poses as described in Section 5.3.1 with an reference upper body size of $s_{ref} = 100$. The RMS consistently improves for all swimming styles. Compared to previous work in [113], RMS values for all styles but breaststroke improved.

### 5.4.2 Key-Pose Retrieval

The initial motivation for improving the joint predictions of swimmers was to similarly improve the identification of specific key-poses. Therefore, we use a SSD [72] detector to track the swimmers in all videos of the *mmc-swim-channel* dataset and estimate the pose using a standard CPM which was fine-tuned on the image sequences of the dataset. We then apply the whole rectification pipeline and extract poses using both the naive approach N-TS and the Random Forest key-pose classifier RF-TS from the previous chapter for key-poses identification. In the following,

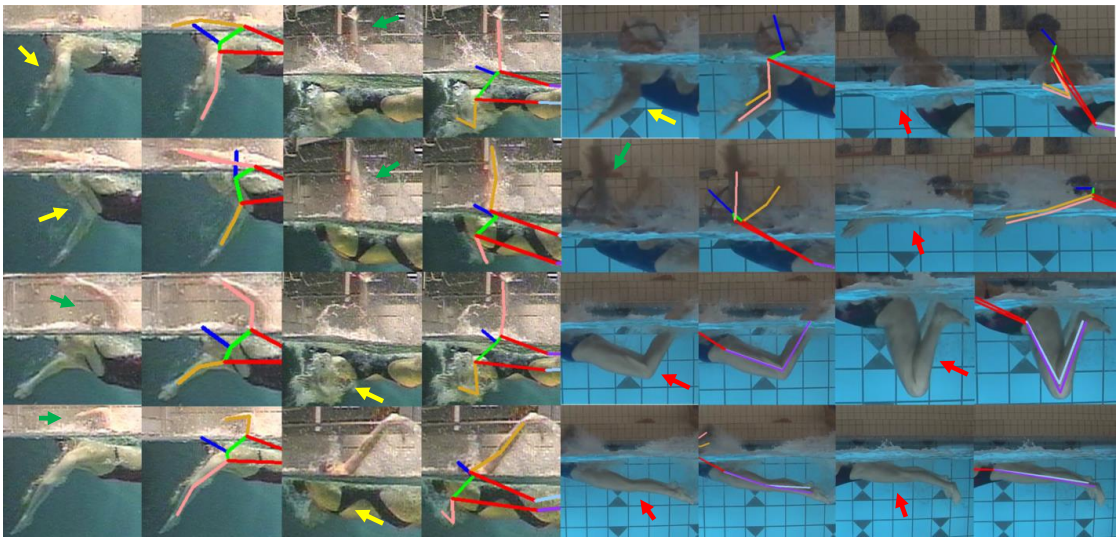


Figure 5.7: Example of key-poses defined by human experts. Raw images are depicted to the left of rectified pose estimates. From left to right: Freestyle, backstroke, butterfly, breaststroke. Arrows indicate the pose-features: upper arm angle $\pi/2$ (green), upper arm angle $3/2\pi$ (yellow), angle between two parts (red).

N-TS is applied to the raw CPM output and serves as the baseline which is compared against the pose rectification pipeline.

Compared to the subset of data evaluated in the last chapters, the complete dataset *mmc-swim-channel* includes much more videos, different swim styles, and new key-pose annotations. Therefore, the results we report in the following are not comparable to scores from the previous chapters.

As a substitute, we compare the performance of the pipeline against one of our older approaches discussed in [113]. In this publication, we used a stricter metric where a key-pose prediction is counted as a True Positive (TP) if a ground truth annotation lies within $\pm 6$ frames instead of $\pm 10$ frames, and no other detection is closer to the ground truth. All detections that can't be matched against a ground truth are considered false positive detections (FP). If a ground truth is not matched against a key-pose prediction, we count it as a False Negative (FN). Figure 5.7 depict the key-poses which we evaluate for the new dataset. The arm angles evaluated for freestyle swimmers are the same for backstroke swimmers. The same angles are evaluated or backstroke swimmers. Both symmetrical swim styles include the pose features "closest and widest angle between the upper and lower arm and leg".

| | free | back | fly | breast | mean |
|---|---|---|---|---|---|
| PCKF CPM baseline (N-TS) | 0.791 | 0.898 | 0.820 | 0.852 | 0.840 |
| PCKF Zecha et al. [113] (N-TS) | 0.828 | 0.920 | 0.851 | **0.901** | 0.875 |
| PCKF CPM refinement N-TS | 0.830 | 0.925 | **0.853** | 0.898 | **0.876** |
| PCKF CPM refinement RF-TS | **0.838** | **0.928** | 0.801 | 0.817 | 0.846 |
| Pr CPM baseline (N-TS) | 0.781 | 0.773 | 0.874 | 0.837 | 0.816 |
| Pr Zecha et al. [113] (N-TS) | 0.851 | 0.823 | **0.891** | **0.876** | 0.860 |
| Pr CPM refinement N-TS | **0.859** | 0.827 | 0.890 | **0.876** | **0.863** |
| Pr CPM refinement RF-TS | 0.858 | **0.838** | 0.811 | 0.793 | 0.825 |

Table 5.4: PCKF and Precision of key-pose retrieval. We compare the baseline established by a fine-tuned CPM against our system from [113] and the rectification approach proposed in this chapter.

PCKF values for the whole dataset are summarized in Table 5.4. Within the human error of ±2 frames, we report considerable PCKF increases between +2% to +5% for all swimming styles. On average, rectification with N-TS can be improved by 3.6% compared to the baseline. Compared to [113], we achieve slightly better scores. Confirming the results from the previous chapter, the random forest classifier works well for anti-symmetrical swimming styles where only arm angles define the pose features. Surprisingly, the average PCKF for both symmetrical swim-styles drops to 81.7% and 80.1%, respectively, which even lies below the baseline performance when using N-TS functions. The reason for this is the pose feature "closest angle" or "widest angle" between upper arm and forearm or between the thigh and the lower leg. In the naive approach, functions for these pose features are deliberately designed to be maximal when the feature occurs. As the random forest features were designed without temporal attributes in mind, the RF-TS model is not able to these pose features well. This finally leads to inferior PCKF scores.

We additionally report the precision for all retrieved key-poses given the inlier threshold of ±6 frames around a key-pose. An increase of up to 7% for the precision means that we not only are able to improve on correctly predicted key-poses but also retrieve far less false positive predictions for key-pose occurrences. Similar to the PCKF drop for RF-TS key-pose estimation, we can also observe a drop in precision for this classification scheme.

## 5.5 Summary

In this chapter we demonstrated that a joint tracking and rectification pipeline can improve estimates of an athlete's joints and thereby considerably increase PCKF and precision of retrieved key-poses. The proposed pipeline consists of a joint-kinematic based integer linear program which is used to identify and correct joint swaps, a simple scheme for diminishing directional residual offset and a robust regression approach for replacing outliers with motion-consistent joint landmarks.

Contrary to other work in this field, the discussed approaches offer several novelties:

- The ILP constraints explicitly model the possibility that joints may change their label.

- Edge weights in the LP-optimization problem strictly follow motion kinematics. This entails that multiple transitive edges connecting the same detections have to be defined and mutually excluded in the constraint set. Other approaches like [59] use intricate neural networks to determine edge weights, which introduce another potential error source to the problem.

- Obtaining large quantities of high precision annotations for training sophisticated machine learning algorithms remains a time-consuming and tedious challenge. Especially in less popular sports, this effort is often not feasible. The proposed pipeline for improving continuous pose predictions can be trained with a comparatively small set of annotated training data.

Experiments illustrate that there still is room to improve on both joint estimation and key-frame extraction. For example, partner graph optimization is often challenged if partner joints occlude themselves for longer time periods, like hips or knees of swimmers. Improving long term tracking of joints remains future work.

# 6 — Jump Dynamics Prediction of Ski Jumpers

We address the problem of automatically predicting the jump forces of ski jumpers in this chapter. The motion sequence of a ski jump can be divided into four phases depicted in Figure 6.1. During the *in-run*, the athlete accelerates while skiing down a jumping ramp. Reaching the end of the ramp, which is denoted the *take-off table*, the athlete launches himself through a jumping motion into the *flight phase*. During the flight phase, the jumper targets to assume an optimal posture to increase flight time and thereby extend jump distance. The jump is completed with the *landing phase*, where the final distance is rated by referees to yield a final score for the jump.

The final jump distance can be actively influenced by the athlete during each phase of the jump. An optimal streamlined in-run guarantees enough initial velocity, while a steady and precisely executed jump motion on the take-off table and a stable and optimal flying posture increase the final distance. During training sessions, coaches put much effort into optimizing flight posture and the ideal timings for executing the jump motion on the take-off table. Therefore, ski jump hills are equipped with several cameras in parallel with the flight path of the jumper. Additionally, force measurement plates are installed in the jump-off table to precisely measure the jump force during the last meters of the in-run. After a run, coaches review the camera footage and measured forces. Similar to the kinematical evaluation, the jumper's posture is manually annotated and the force measurements are evaluated by identifying points of interest like the temporal occurrence of the strongest force. This whole process is usually done manually and thereby time-consuming.

In this chapter, we strive to automate the jump evaluation process by leveraging the poses of an athlete to estimate the jump force. Therefore, two problems need to be solved. First, different camera views need to be registered in a mutual co-

Figure 6.1: The different phases of a ski jump. From left to right: in-run, jumping, flight phase, landing.

ordinate system, followed by temporal synchronization between camera views and force measurements (Figure 6.2). Second, a model for generating sequences of force predictions from sequences of pose estimates has to be defined. The methodology discussed in this chapter was published in [111].

## 6.1 Related Work

**Camera-calibration.** The topic of non-classical camera-calibration, i.e., not based on homography estimation between overlapping cameras, was addressed from different directions in the past. [68, 105] propose search algorithms for temporal alignment using human kinematics and point trajectories in dual camera setups. Padua et al. [79] extend the two camera scenario to multi-camera systems. They define a search space of calibration and synchronization parameters called a timeline and proposed an optimization algorithm for approaching the optimal settings. A common feature in all these algorithms is that cameras need to film the same scene to optimize spatiotemporal parameters.

**Object dynamics/kinematics.** Predicting dynamics of objects in still images is commonly based on neural networks with two input arms, one for a (masked) image and a second for force inputs. Forces are modeled using a game engine [75] or a physics engine [76, 107], where the whole image scene can be simulated. Alternatively, scenarios like billiard games [13, 38] are examined, where a force can be applied to the scene in a controlled environment. Recurrent networks have mainly been used for the prediction of human dynamics from just image/video. For example, deep recurrent neural networks(RNNs) [74] and Encoder-Recurrent-Decoder networks [39] are used to predict the motion of a person continuously. While

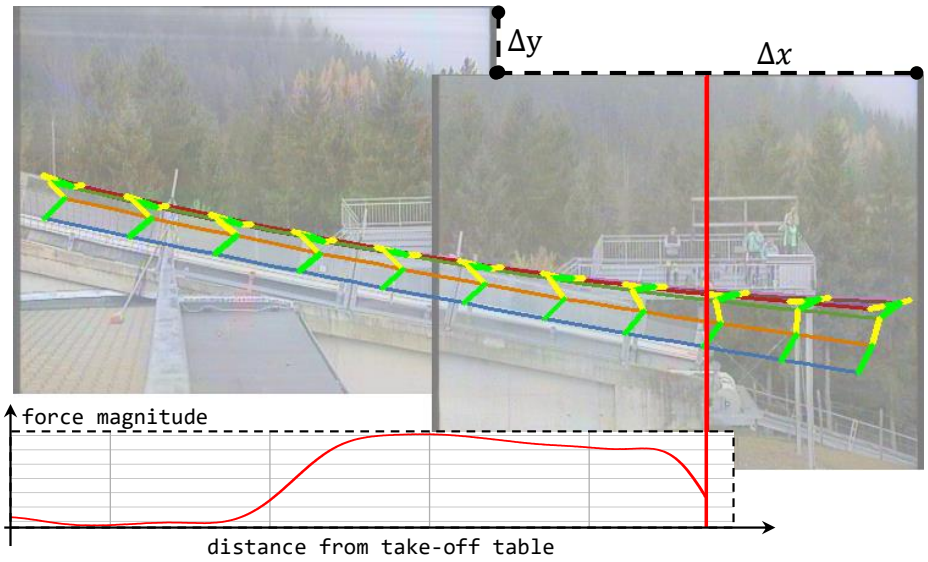



Figure 6.2: Based on the predicted poses of an athlete (green/yellow), joint trajectories (blue, orange, dark red) and camera offset $\Delta x$ and $\Delta y$ for two camera views can be inferred with the method described in Section 6.2.2. The red line indicates the edge of the take-off table, which serves as the anchor for force-image synchronization. The synchronized force signal for the jump is depicted as a red graph.

they often state that the dynamics of a person's movement are estimated, the actual prediction only entails joint kinematics.

**Sequence to sequence learning.** Sequence to sequence networks have been extensively researched in the last years for machine translation [64], audio synthesis [102] or language modeling [22]. While network structures build on Recurrent Networks [10], LSTMs [54] or Gated Recurrent Units [17] are numerous, convolutional sequence to sequence modeling as an alternative has gained traction recently in machine translation [40] as well as human kinematics prediction [69]. Compared to recurrent models, the width of the history size can be directly defined and controlled through the receptive field of the network. Additionally, computations over all layers can be fully parallelized (in contrary to recurrent models).

**Our work.** While our work is strongly influenced by generic state-of-the-art networks [7] for time series prediction with convolutional networks, classical applications of these models often work on discrete-valued input and output data. In contrast to that, we attempt to use continuous input data to predict purely continuous output data. The prediction of real-world force measurements solely form

camera footage in an entirely uncalibrated scenario - especially within our scope of application, is a novel problem.

## 6.2 Multimodal Registration and Synchronization

The process of predicting forces from human poses involves the registration and synchronization of different modalities. We describe the measurement setup in the following and propose a calibration strategy for registering different cameras in the system and temporally synchronizing them with force measurements.

### 6.2.1 Measurement Setup

A ski jump hill is equipped with $N$ cameras $\mathbf{C} = (C_1, ..., C_N)$ along the take-off table and the flight trajectory of the athlete. All cameras shoot at 50 Hz and are temporally synchronized by the athlete triggering a light barrier when he starts his jump at the top of the in-run. Cameras are regularly spaced in parallel to the fight path and usually do not overlap, except for two cameras with minimal overlap pointing at the take-off table. While all image planes are assumed to be parallel to the flight trajectory of the athlete, we can not guarantee the same distance between each image plane and the jumper due to terrain limitations.

Force measurements are taken at a frequency of 2 kHz from a series of force measurement plates installed at the last 17 meters of the in-run, up to the edge of the take-off table. An external synchronization system for cameras and force plates does not exist. The plates are triggered and synchronized among one another by several light barriers along the in-run. The exact process of measuring the jump forces over the last 17 meters of the ski jump is unknown to us. The output of the measurement system is a series of force readings sampled at a rate of 2 kHz. Each reading is labeled with the distance between the ski jumper and the edge of the ski jump. As a result, the position of the jump edge together with the location of the jumper defined by his ankle joint can be used as a temporal synchronization anchor in both force signal and continuous pose estimate, respectively. Using the position of the light barriers directly is not possible as they are typically not visible in the camera images.

### 6.2.2 Pose Estimation for Robust Camera Calibration

We propose to use post estimates of an athlete for image registration and make some simplifying, but reasonable assumptions about the camera setup. Firstly, the image planes are approximately parallel to the flight path of the jumper and they are horizontally aligned with the horizon. While we found both assumptions to be true in our setup, incorrect installation of the cameras could entail a rotation in the image plane. In this case, parameters for camera rotation have to be added to the proposed alignment model. Secondly, the detector used for pose estimation returns mostly decent pose estimates, although a small set of outliers should not distort the estimated model parameters. Thirdly, the flight path of the athlete between two camera views can be approximated with a second-degree polynomial.

#### Pose Preprocessing

Under these assumptions, we first use a MobileNet [85] to detect a ski jumper in all images in every video of a video set. The pose of the jumper is estimated using the CPM-VGG network introduced in Chapter 5. Both the detection network and the pose estimator are fine-tuned using annotated jumper poses. The output of the pose estimator is a set of $J$ joint coordinates for the head, shoulder, elbow, wrist, hip, knee and ankle of the body side facing the camera. We found that the PCK of our fine-tuned pose estimator (Figure 6.6) is on a par with state-of-the-art pose estimation systems, if not even better due to the limited application domain and the uniformity of jumper poses. Within the acceptable threshold of 0.2 of the upper body length of an athlete, we can detect >99% of all joints correctly. Even with a more restrictive threshold of 0.1, we still get a PCK of 95%.

As indicated in Section 6.2.1, we have to account for different camera scales. For all poses obtained from one camera view $C_i$, we compute a scaling factor $s_i$ based on the mean upper body size $s_{upper}$ of the athlete, which is defined as the distance between shoulder and hip, in that shot:

$$s_i = \frac{s_{ref}}{s_{upper}}.$$

$s_{ref}$ denotes a reference body size. We rescale all pose estimates in the $i$-th camera view with $s_i$, resulting in the athlete having approximately the same relative size in all camera views. We found that the upper body size hardly changes within

a camera view, can be detected very robustly and is, therefore, a good choice for obtaining a robust scale for each camera. To get an even more robust estimate for the camera view scale, we exclude 10% of the largest and smallest detected upper body length when computing the mean in one view.

### Overdetermined Trajectory System

From our initial pose estimates of the ski jumper in all camera images of one run, we derive a robust polynomial regression for the trajectories of all joints of the athlete as follows. Let again the location be defined as $\mathbf{l}_j^{(t)} = [\hat{x}_j^{(t)}, \hat{y}_j^{(t)}]^T$, where $\hat{x}_j^{(t)}$ and $\hat{y}_j^{(t)}$ are the two dimensions of the joint coordinate of joint $j$. A robust approximation of the joint trajectory and the camera offsets $\Delta x$ and $\Delta y$ for each location dimension is obtained using a $2^{nd}$ degree polynomial. For the $x$ dimension, it is given by

$$\hat{x}_j^{(t)} = \begin{cases} [\mathbf{t}\ \ 0]^T \cdot [\mathbf{b}_j\ \ \Delta x] & \text{if } \mathbf{l}_j^{(t)} \in C_i \\ [\mathbf{t}\ \ 1]^T \cdot [\mathbf{b}_j\ \ \Delta x] & \text{if } \mathbf{l}_j^{(t)} \in C_{i+1} \end{cases} \tag{6.1}$$

where $\mathbf{b}_j = [b_{j,2}\ \ b_{j,1}\ \ b_{j,0}]$ are the coefficients of the polynomial, $\Delta x$ is the offset in x direction and $\mathbf{t} = [t^2\ \ t\ \ 1]$ is a vector of polynomial variables. In order to estimate the parameters $\mathbf{b}_j$ and camera offset $\Delta x$ for two adjacent camera perspectives $C_i$ and $C_{i+1}$ using all $J$ joints and all $K$ detected poses from both views, we define an overdetermined system of equations

$$\hat{X} = AB. \tag{6.2}$$

Here, $\hat{X} \in \mathbb{R}^{KJ}$ is a vector of detected coordinates for each joint from each pose estimate. The parameter vector $B \in \mathbb{R}^{3J+1}$ is comprised of all polynomial coefficients and the offset, i.e,

$$B = [\mathbf{b}_1 \dots \mathbf{b}_J\ \ \Delta x].$$

Each row in the sparse design matrix $A \in \mathbb{R}^{KJ \times (3J+1)}$ represents the polynomial variables for one joint detection (time the joint was detected) and is therefore connected to exactly one target value $\hat{x}_j^{(t)} \in \hat{X}$. As the structure of this matrix is a bit intricate to write down, we illustrate the composition of one row in $A$ with a descriptive example. Assume that at time $t = 4$, we detected the shoulder joint $j = 1$ at x position 356 in camera $C_{i+1}$. Then there would be a row $\mathbf{a}_i$ in matrix $A$

coding this detection as

$$\mathbf{a}_i = [\overbrace{0\ 0\ 0}^{j=0}\ \overbrace{t^2\ t\ 1}^{j=1}\ 0 \dots 0\ \Delta x] = [0\ 0\ 0\ 16\ 4\ 1\ 0 \dots 0\ 1]$$

and the corresponding target value in $\hat{X}$ would be $\hat{X}_i = \hat{x}_1^{(4)} = 356$. Note that according to Equation 6.2, this row of $A$ is multiplied by the parameter vector $B$. Therefore, the sparsity of this vector guarantees that only relevant parameters affect the robust estimation of the joint trajectory. The last attribute in row vector $\mathbf{a}_i$ is set to 1 because this sample was taken from camera $C_{i+1}$. It would be = 0 if this specific joint were predicted in camera $C_i$. Consequently, the estimated parameters express all trajectories relative to the reference camera $C_i$. While the above deviation was only defined for $\hat{x}_j$, the approximation of $\hat{y}_j$ is correspondingly defined according to Equation 6.1 and solved with a separate system.

**Robust Parameter Estimation**

An optimal least-squares solution to Equation 6.2 is given by $B = (A^T A)^{-1} A^T \hat{X}$. However, we have to account for inaccuracies like detection outliers in the sequence of predicted poses. Hence, we choose to minimize absolute L1-norm deviations. A set of robust trajectory parameters and camera offsets $B^*$ can be obtained by minimizing

$$B^* = \arg\min_B = ||\hat{X} - AB||_1 \tag{6.3}$$

A solution to this problem can be robustly approximated using the iterative method in Algorithm 1. This procedure is applied to all pairs of adjacent camera views. The resulting trajectories and camera offsets for the two cameras pointed at the take-off table are visualized for one video in Figure 6.2 for one camera pair. The determined trajectory parameters $\mathbf{b}_i$ are used to resample the joint positions to the sampling rate of the force plates (2 kHz). Thereby, one pose estimate can be assigned to each measured force value.

### 6.2.3 Synchronization of Force and Camera

With all camera images spatially registered, we are left with temporally synchronizing camera views and measured forces. As stated previously, we know for each force value how far it was taken from the edge of the take-off table. Together with

the upsampled pose estimates from the spatially registered camera views, temporal synchronization between both modalities is straightforward: The edge of the table is manually annotated in the respective camera view. The temporal synchronization anchor in the camera views is determined as the point where the ankle of the ski jumper passes the edge and consequently leaves the last force plate. The synchronization point for the series of force measurements is merely the last data point where a force measurement was taken on the take-off table. Both modalities are aligned at both temporal anchors. Finally, we crop the series of pose estimates and the synchronized force measurements to the same length of 1000 samples, equivalent to the last 25 frames of the jumper before take-off.

## 6.3 Temporal Convolutional Model

We present a temporal convolutional sequence to sequence model based on the findings of Bai et al. in [7], who propose a basic architecture for a temporal convolutional network (TCN) which aims at combining the best recent practices in convolutional network design for sequence prediction. TCNs were originally designed to map a sequence of input length $n$ to an output sequence of the same length using only causal convolutions, i.e., there is no "leakage" from future values to the past. TCNs aim to build a long effective history size, meaning that they are able to look very far back into the past to make a prediction. Most commonly, TCNs are used for word-level language synthesis, machine translation modeling or polyphonic music modeling [22, 64, 102] and are tested on benchmark tasks defined on discrete valued input and output data. For our application, we modify the basic TCN blocks and propose a network for predicting a continuous time series from a continuous pose input.

### 6.3.1 Dilated Convolution Blocks

A major disadvantage of modeling a sequence to sequence network only with standard convolutional layers is that a very deep network with large kernel sizes has to be constructed in order to obtain a wide receptive field and thereby a large history size. Training very deep networks with large kernels from scratch is often not feasible, especially if comparatively few training data is available. Recently, dilated convolutions have been extensively researched for increasing the receptive field of a

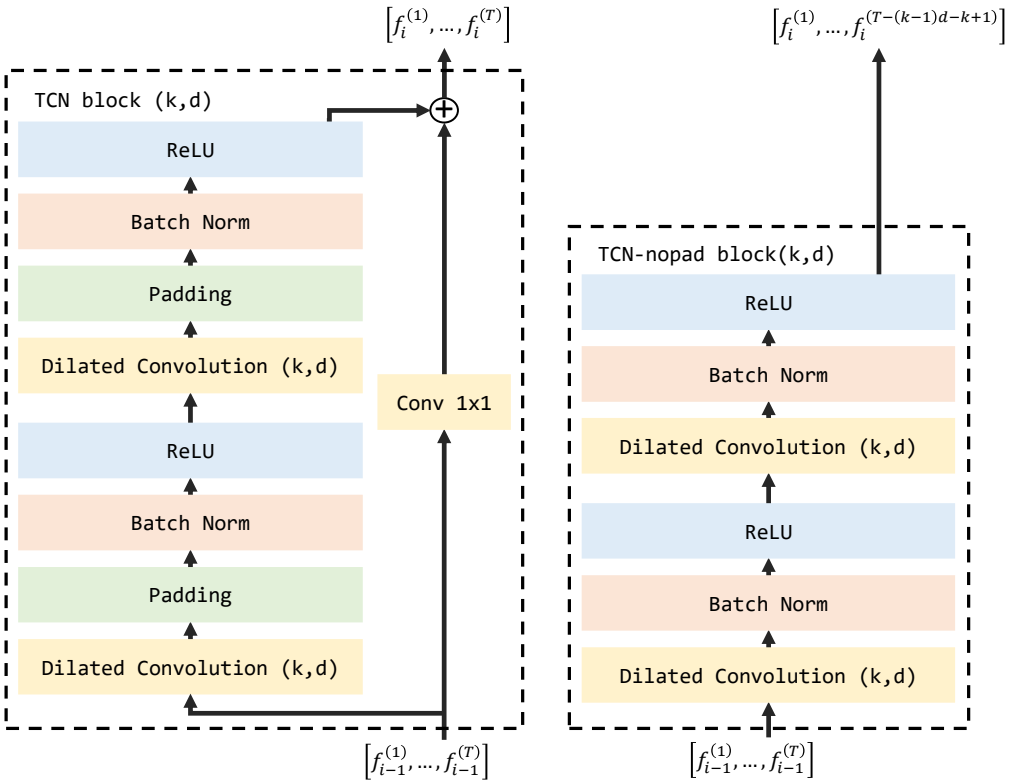



Figure 6.3: Left: standard TCN block inspired by [7]. Right: TCN block without padding and residual connection.

convolutional layer with a small kernel without having to add additional layers. A dilated convolution $*_d$ on an arbitrary input sequence $g \in \mathbb{R}^l$ of length $l$ is convolved with a kernel $h \in \mathbb{R}^k$ of size $k << l$ using a dilation factor $d$ through

$$(g *_d h)[t] = \sum_{i=0}^{l-1} g[t - id] \cdot h[i] \tag{6.4}$$

Dilated convolution spreads the filter coefficients over a larger area of the input signal, increasing the receptive field and thereby - in a temporal context - expands the history of past values.

Inspired by [7], we define a temporal convolution network block (TCN block, dubbed as *tcn* in the experiments) as the basic building block for our networks. Such a block is depicted in Figure 6.3. It includes two consecutive dilated convolutional layers with additional padding, augmenting the output features to the same size as the input features. Each convolutional layer is followed by batch-normalization [57]

and a Rectified Linear Unit [77]. A residual connection [53] is added to the output of the block, which allows for learning only the difference between the input and the output. Each TCN block has two parameters $k$ and $d$ for the kernel and dilation size, respectively.

In the standard definition of the TCN block, the length of the input signal does not change throughout the network due to extensive padding. However, we found that there is a drawback to dilated convolutions which arises especially when the dilation factors get very large. In this case, the effective size of the kernel is very large and the amount of zero-padding has to be equally increased to ensure that the output signal has the same length as the input signal. For large dilation factors, e.g., $d > 10$ at kernel size $k = 5$, extensive padding introduces a lot of volatility at the edges of the final output signal (see Figure 6.7). Consequently, the gradients produced from fluctuating forces at the edges of the output signal may introduce a spurious training stimulus for all underlying convolutional kernels.

To examine this effect, we define a variation of TCN block which is also depicted in Figure 6.3 and will be dubbed *tcn_nopad* in the experiments. This block waives the padding operation after each convolutional layer. Consequently, the skip connection has to be dropped due to the difference in input and output size of the block. We refer to a TCN block without padding as tcn_nopad. The output shapes of the last TCN-output layer are listed in Section 6.4.

### 6.3.2 Network Structure and Losses

The overall structure of a convolutional sequence to sequence network is built on TCN blocks (*tcn* or *tcn_nopad*). A sample architecture is depicted in Figure 6.4. The input of the network is a time series of pose estimates as described in Section 6.2.2. The input layer is followed by a number of TCN blocks. The first TCN block after the input layer performs a $k \times 14$ convolution with kernel size $k$ and 14 x and y coordinates of the seven annotated joints. In order to increase the receptive field of the network quickly without having to add too many layers, the dilation factor for block $q$ in the sequence is set to be $d = 2^q$. In our experiments, the depth of the network does not exceed 7 TCN blocks. We additionally experiment with different quantities of kernels per block and compare classical design approaches based on a steadily increasing number of kernels with strategies where the amount of kernel first increases and then decreases. Details of different network parameters are listed

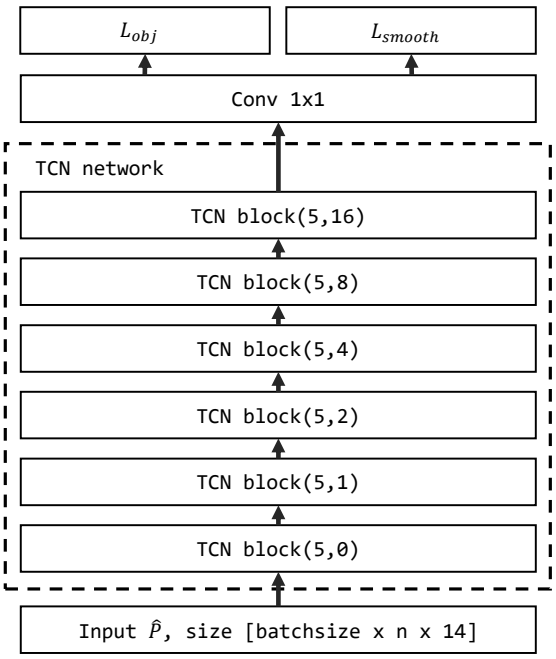



Figure 6.4: An exemplary network architecture comprised of 6 TCN blocks (tcn_6)

in Table 6.1. A final convolutional layer terminates the network with kernel size 1 and linear activation, which reduces the output of the last layer to a one-dimensional sequence of force values.

The training loss of our convolutional sequence to sequence networks is defined by three partial losses. First, the objective loss between the ground truth force $F$ and the predicted force $\bar{F}$ is defined by the piece-wise Huber-Loss with $\delta = 1$, defined as

$$L_{obj}(F - \bar{F}, \delta) = \begin{cases} 0.5(F - \bar{F})^2 & |F - \bar{F}| \leq \delta \\ \delta(|F - \bar{F}| - 0.5\delta) & \text{otherwise.} \end{cases}$$

Second, in order to avoid overfitting convolutional layers, a $L_2$-regularization loss per layer is introduced. All regularization losses are aggregated in the overall regularization loss $L_{reg}$. Third, we found that the output of the network is often noisy. In order to encourage smoothness in the final force predictions, we define a smoothness loss $L_{smooth} = L_{obj}(\frac{d}{dt}F - \frac{d}{dt}\bar{F}, \delta)$ to be the Huber-Loss of the difference between derivatives of ground-truth and prediction forces. The derivative of either time series is simply determined numerically by $\frac{d}{dt}F(t) = F(t) - F(t-1)$.

The total network loss is defined as the weighted sum of all partial losses:

$$L_{total} = L_{obj} + \alpha L_{reg} + \beta L_{smooth} \tag{6.5}$$

with $\alpha$ and $\beta$ being loss weights.

## 6.4 Experiments

The prediction of continuous dynamic parameters of ski jumpers (and athletes in general) solely from continuous pose estimates using convolutional sequence models is a novel problem in the scientific community with no published results yet. Hence, we focus on analyzing different network structures quantitatively and qualitatively.

### 6.4.1 Dataset

We use the dataset described in Section 2.3.2 for training the ski jumper detector and pose estimator. The task of predicting jump forces is analyzed on a subset of 225 video set of different ski jumpers and corresponding force series. We split 180 videos for training different TCN architectures, 20 for validation and 25 for testing. As described in Section 6.2.2, we estimate all poses in each video and derive calibration and synchronization parameters and therefore obtain one sequence of 1000 human poses and force measurements per video. We artificially augment the training set by cropping sequences of length 950 and a stride of five from the original 1000 measurements, which increases the dataset size by a factor of 10. For each joint in the cropped pose series, we add a minimal amount of Gaussian noise with variance $\sigma^2 = 1$ to all joint coordinates. This does not have much influence on the robust trajectories, but add spatial variance and therefore more variability. During training, all sequences in the training set are randomly shuffled.

All force series are normalized by the largest occurring force value in the dataset, which maps all measurements into the interval $[0, 1]$. For *tcn_nopad* networks, we resample the force time series to the proper network output size using FFT-resampling from [63].

### 6.4.2 Error Measures

We report the Mean Squared error (MSE) and median of the squared errors between the ground truth $F[t]$ and the predicted forces $\bar{F}[t]$ for different network structures. All errors are computed over all test-videos(see Table 6.2). In order to give a better intuition of network performances, we additionally try to answer a real-world question asked by coaches during the evaluation process: At which time in the video is the maximal take-off force observed? In a training situation, the corresponding frame depicting the distance between athlete and edge of the jump-off table gives the coaches a visual clue for adjusting the timing of the jump in the next run. For measuring how well we are able to identify this frame, we use the PCKF measure from the previous chapters again.

Therefore, we take the argument of the maximal predicted force value $t = argmax_t(\bar{F}[t])$ and compared it to the maximal force occurrence of the ground-truth. By identifying the corresponding frames for both maximal forces, we can use the PCKF measure, which counts a predicted maximal force occurrence as correct if the temporal distance to the ground-truth frame falls within a threshold $\tau$. In the case of unpadded networks (tcn_unpad), the output sequence length $s$ is implicitly defined by the number of dilated convolutional layers and the filter size. Hence, we use the scaling factor $950/s$ to rescale the offset between prediction and ground-truth in the short output sequence, allowing for comparing it with offsets in signals with original length of 950 force values.

### 6.4.3 Tested Network Structures and Training Parameters

With the novelty of the proposed application, specifically in the context of the convolutional sequence to sequence networks, a state-of-the-art for network architectures and parameters is non-existent. We see this as an opportunity to perform experiments on a variety of networks, summarized in Table 6.1. Three different network types are examined. The basic *tcn*-network is implemented with four to six consecutive tcn blocks (*tcn_4*, *tcn_5* and *tcn_6*). Non-padded versions (*tcn_unpad_4*, *tcn_unpad_5* and *tcn_unpad_6*) are used to examine the influence of dilation on volatile output sequences. All networks first increase the number of kernels up to the middle of the network, then decrease it again. We compare this parameter set to two networks *tcn_5_wide* and *tcn_6_wide*, which implement a more traditional

| name | len. input | len. output | #blocks | #channels/block |
|---|---|---|---|---|
| *tcn_6* | 950 | 950 | 6 | [32,64,128,64,32,16] |
| *tcn_5* | 950 | 950 | 5 | [32,64,128,64,32] |
| *tcn_4* | 950 | 950 | 4 | [32,64,128,64] |
| *tcn_nopad_6* | 950 | 830 | 6 | [32,64,128,64,32,16] |
| *tcn_nopad_5* | 950 | 702 | 5 | [32,64,128,64,32] |
| *tcn_nopad_4* | 950 | 446 | 4 | [32,64,128,64] |
| *tcn_6_wide* | 950 | 950 | 6 | [32,64,128,128,128,128] |
| *tcn_5_wide* | 950 | 950 | 5 | [32,64,128,128,128] |
| *tcn_7* | 950 | 950 | 7 | [32,64,128,128,64,32,16] |
| *tcn_2_shallow* | 950 | 950 | 2 | [64,128] w/ dilation [2,16] |

Table 6.1: Overview over different tested network architectures. *tcn_x* networks are build from standard tcn blocks, *tcn_nopad_x* nets discard padding. *tcn_x_wide* architectures increase the number of kernels from layer to layer.

distribution of kernels, i.e., the number of kernels is steadily increased throughout the network. We also experiment with two more exotic networks, namely *tcn_7* and *tcn_2_shallow*. *tcn_7* is a 15 layer network, i.e., seven consecutive TCN blocks and the final layer, where the receptive field covers the entire pose input of the network. *tcn_2_shallow* is a shallow net comprised of 2 TCN blocks. The dilation factors for both blocks in this network were set to 2 and 16, respectively. With this network, we want to examine if the reconstruction of a force signal can be achieved with very few layers with large dilation factors. An overview of all network parameters is given in Table 6.1.

The layers in all networks are initialized with variance scaling initialization [52] and trained until convergence in mini-batches of size 16 with a learning rate of 0.01 and a learning rate reduction by a factor of 10 every 3 epochs. The loss weights for the regularization loss was set to $\alpha = 0.0001$, while the smoothness loss was set to $\beta = 1$

### 6.4.4 Results

Before presenting the force prediction results for all networks, we present a short discussion for the joint PCK obtained using the CPM-VGG variant of the CPM network.

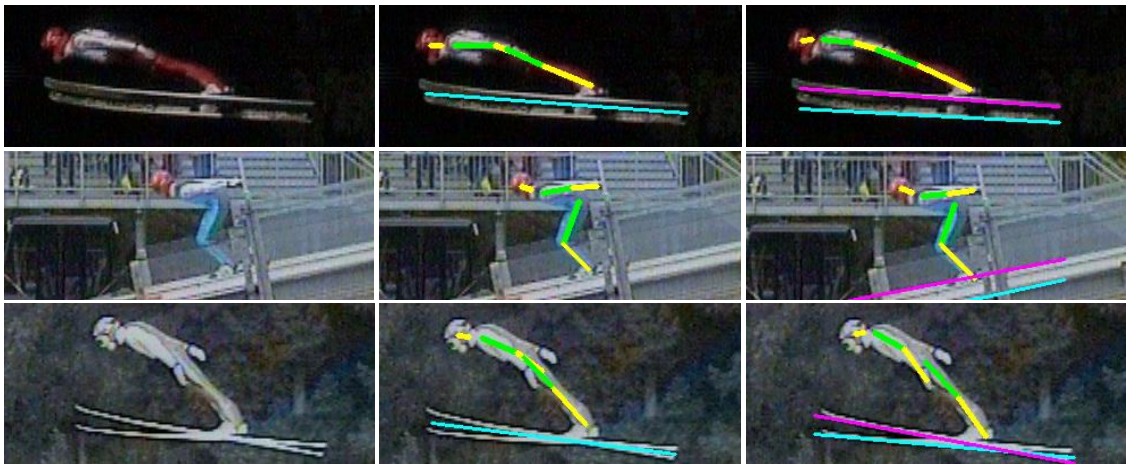


Figure 6.5: Examples of pose estimates predicted by a fine-tuned CPM-VGG network. Left: original image. Middle: expert annotation. Right: CPM-VGG output.

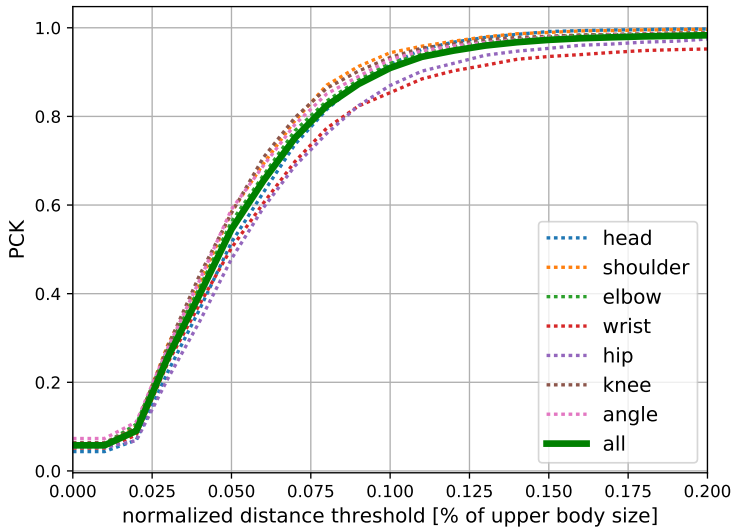



Figure 6.6: Percentage of correct keypoints [86] for the fine-tuned CPM-VGG used for estimating the pose of the jumper. We achieve very competitive results (PCK@0.2>0.99), which allows for on a par pose estimates as a good starting point for camera calibration and force prediction.

### Joint PCK

We found that the PCK of our fine-tuned CPM-VGG network, depicted in Figure 6.6), is on a par with state-of-the-art pose estimation systems. Within the acceptable threshold of 0.2 of the upper body length of an athlete, we can detect >99% of all

| Network | MSE | median |
|---|---|---|
| tcn_6 | **0.071** | 0.049 |
| tcn_5 | 0.074 | 0.051 |
| tcn_4 | 0.088 | 0.060 |
| tcn_nopad_6 | 0.105 | 0.067 |
| tcn_nopad_5 | 0.087 | 0.057 |
| tcn_nopad_4 | 0.091 | 0.061 |
| tcn_6_wide | **0.071** | 0.049 |
| tcn_5_wide | 0.073 | **0.047** |
| tcn_7 | **0.062** | **0.043** |
| tcn_2_shallow | 0.093 | 0.065 |

Table 6.2: MSE and median of squared errors between ground truth and predicted forces.

joints correctly. Even with a more restrictive threshold of 0.1, we can get a recall of 95%. Figure 6.5 depicts detection results.

### MSE and prediction quality

The results of our experiments are listed in Table 6.2. We find that dropping the padding does not entail good predictions regarding MSE. All architectures without padding performed considerably worse in all experiments. A visual inspection of some outputs in Figure 6.7 (middle column) supports the quantitative errors. All *tcn* networks and their *tcn_wide* variants seem to perform much better. MSE errors indicate that more layers in a network result in superior performance over networks with a smaller receptive field. Comparing the quality of the best performing networks *tcn_6* and *tcn_6_wide* (left two columns in Figure 6.7), we see that *tcn_6* produces more volatile predictions at the edge of the signals, despite them having the same MSE and median error. In general, the observed volatility is a product of the large dilation factors and stems from the convolution of a filter with large amounts of padded zero-values. The prediction noise can be observed in all padded *tcn* networks. From our experiments, we conclude that adding more filters to the networks, especially in later layers, might improve force predictions at the edges of the output signal.

The evidence for the hypothesis that deeper networks are superior in reproducing the force signal is supported by the error of *tcn_2_shallow*, which performs similarly

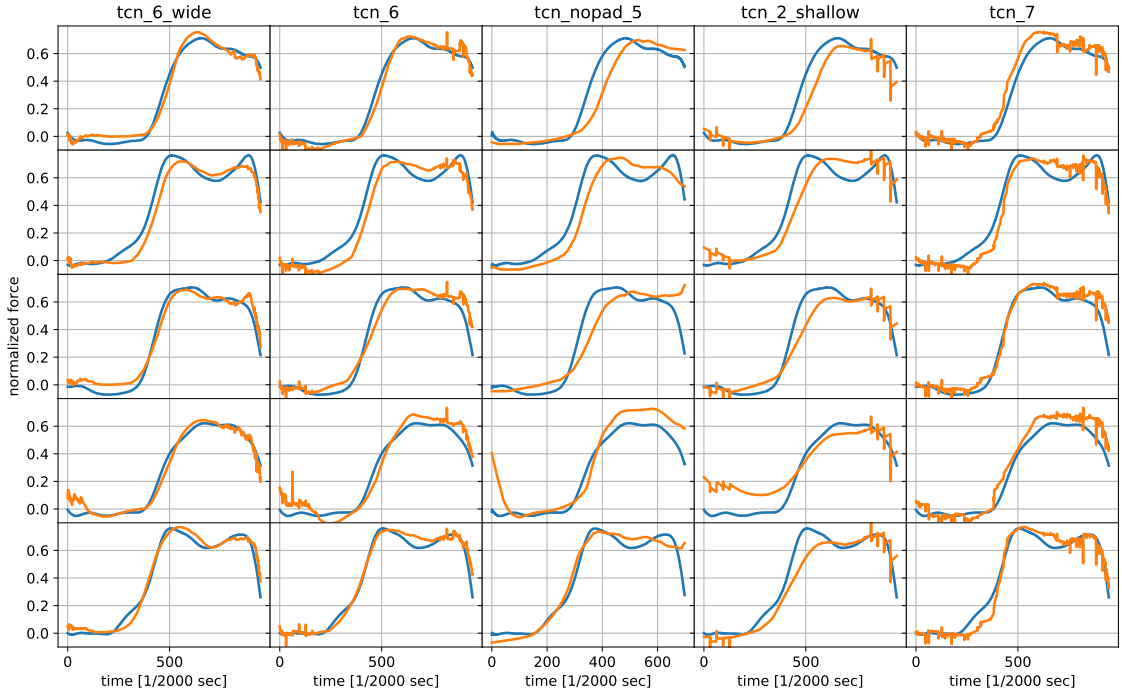



Figure 6.7: Qualitative outputs of different architectures. The best performing networks in terms of MSE (first three columns) and two more exotic variants (last two columns) are compared for five different force series (rows). Ground-truth measurements are depicted in blue, predictions in orange.

to the *tcn_nopad* versions. While this network grasps the general trend of a force graph (fourth column in Figure 6.7), a significant amount of volatility can be observed at the edge of the predicted signals, despite a large number of filters in this model.

The best results are obtained by the deepest network *tcn_7*. As previously stated, the perceptive field of this network covers the complete input. While it produces a relatively small MSE, we can also observe in column 5 of Figure 6.7 that it creates more outliers in the predicted output than all other architectures. While the overall performance of *tcn_7* may be favorable, the outlier problem has to be addressed to deliver a visually superior prediction.

### PCKF

We conclude our experimental section with a short discussion of the PCKF performance of all models, depicted in Figure 6.8. Our best models *tcn_6* and *tcn_6_wide* detect 25% of the frames depicting the jumper during the maximal leg force mea-

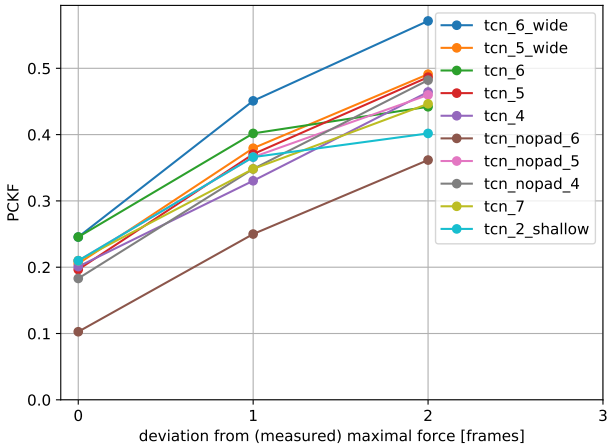



Figure 6.8: Percentage of correctly identified key-frames depicting the moment of the maximal leg pressure before the jump. The best-performing network predicts 25% of the frames perfectly correct and 55% within ± 2 frames.

surement correctly. Within a deviation of 2 frames, *tcn_6_wide* - despite producing the same MSE as *tcn_6* - surprisingly produces a superior recall of 55%. Most of the models perform surprisingly well, although no model beats *tcn_6_wide*. It may also be surprising that the best model regarding MSE, *tcn_7*, does not even come close to the best scores. This may be attributed to the fact that we intentionally did not perform any post-processing like low-pass filtering on the predicted force series, at which point the volatility in the *tcn_7* output is detrimental.

On the question of why the recall of the networks is comparatively small in absolute terms, we point to Figure 6.7. We found that force measurements very often have multiple local maxima, sometimes very close together, often with very similar absolute values. This makes the identification of force peaks a great challenge in general, which we will accept in the future.

## 6.5 Summary

In this chapter, we discussed convolutional sequence to sequence networks for the multimodal prediction of jump forces of ski jumpers. The prediction networks use the estimated pose of the jumper to infer a series of force measurements, recorded with force plates at training time. The registration and synchronization between all modalities, i.e., multiple cameras and multiple force measurement plates, was im-

plemented through least squares optimization of the athlete's joint trajectories. The proposed sequence to sequence network structures are strictly convolutional. Similar structures have been used for discrete-valued data problems like an automated translation. We demonstrate that with proper network architecture, a convolutional sequence to sequence model can be used for simple regression tasks on strictly continuous input data. While the experimental results are very promising, problems like the unwanted volatility of output forces have to be solved to produce more competitive results in the future. Furthermore, an interesting question arises when the problem statement is reversed: Is it possible to predict a sequence of poses for a ski jumper with a convolutional sequence to sequence model, given only the measured force values of a jump? This question and many others remain future work.

# 7 — Conclusion

## 7.1 Summary

In this thesis, we propose two general approaches for identifying key-poses of swimmers in a swimming channel.

**Key-Pose Inference from Support-Pose Detections.** Our first approach introduces a spatiotemporal, hierarchical pose detector. It unites an ensemble of poselets, each of which acts as a small detector for discriminative limb configurations. In order to be able to detect different poses within a motion sequence, multiple poselets for the same limb at different temporal stages of the motion are evaluated simultaneously and continuously. The introduced semimetric for clustering limb configurations measures the similarity of poses and allows for clustering pose configurations that are spatially and temporally close. The proposed poselet ensemble is also evaluated hierarchically, i.e., the activation of a poselet in the image is conditioned on the score of a top-level athlete detector. The pose detector is used to identify support poses via the activations of different poselets, which are used in a simple probabilistic scheme to infer key-poses.

**Key-Pose Inference via Articulated Pose Estimation.** Our second approach discusses how key-poses can be directly derived from an articulate estimate of the human pose itself. We introduce a network implementing a classical deformable part model with deeply learned appearance features and additional pairwise relation terms. Two methods for identifying key-poses from a sequence of post estimates are discussed: by querying the pose feature directly using a design function or with a random forest classifier to predict a key-pose without knowledge about the underlying pose feature. We can experimentally show that both approaches perform similarly well for antisymmetrical swim styles, with the classifier beating the intu-

itive method due to being able to adapt the pose estimator's detection bias. For symmetrical swim styles, the random forest classifier performs subpar.

We experimentally show that the second method clearly outperforms the first one. However, this is only the case if the key-pose defining pose feature is clearly measurable within the human pose. If a pose feature is inconspicuous, making the direct detection of the respective (latent) key-pose difficult, the first approach would be suited better.

A closer analysis of novel pose estimators reveals that they suffer from several orthogonal pose estimation errors. The proposed rectification pipeline consists of a joint-kinematic based integer linear program which is used to identify and correct joint swaps, a simple scheme for diminishing directional residual offset and a robust regression for replacing outliers with motion-consistent joint landmarks. We experimentally demonstrate that joint tracking and rectification can improve pose estimates and thereby key-pose identification results.

In the last chapter, we discussed convolutional sequence to sequence networks for the multimodal prediction of jump forces of ski jumpers. The networks use the continuously estimated pose of the jumper to infer a series of force measurements, recorded with force plates at training time. The registration and synchronization between all modalities, i.e., multiple cameras and multiple force measurement plates, was implemented through least squares optimization of the athlete's joint trajectories. The proposed sequence to sequence networks are strictly convolutional and leverage dilated convolutions to keep the network depth small. We demonstrate that with the proper network architecture, a convolutional sequence to sequence model can be used for sequence regression tasks on continuous input data.

## 7.2 Conclusion and Outlook

Throughout this thesis, we focused on the estimation of key-pose occurrences, as they lay the foundation for kinematical evaluation protocols in the real world. It should be emphasized that there is still a lot of potential for improving key-pose estimation of swimmers in a swimming channel. While sophisticated methods for processing time-continuous data, like LSTMs or recurrent networks, show highly promising results in tasks like time series classification and prediction, they share

the commonality of being very data-hungry during training. This requirement is very difficult to fulfill in specific application scenarios like the ones discussed in this thesis. Collecting and preparing large amounts of training data is a tedious process which in our case took many years. Even if data is available, obtaining large amounts of high-quality annotations is a challenging problem, specifically in aquatic environments, where even experts with longtime experience often face great difficulties in identifying specific poses or correct joint placements.

The rectification of estimation errors using optimization strategies faces the same problems. While we are able to perceive that errors are frequent and intricate enough to be a problem, the amount of data necessary to compile a thorough and complete error profile of an estimator, which then can be used to train a sophisticated and reliable rectification approach, is difficult to obtain. With emerging research in the field of 3d pose estimation from single images, an interesting idea might be to include 3d pose estimates into a rectification pipeline, leveraging an additional dimension to enforce constraints on the geometry of the human pose. Of course, it would be preferable not to make estimation errors in the first place. While various approaches for temporal human pose estimation have been proposed over the last years, we still lack the ability - or the amounts of data necessary - to train temporally consistent pose estimators on images sequences directly.

The topic of predicting the jump forces of athletes, in our case ski jumpers, has not been covered exhaustively within this thesis. While experiments show that purely convolutional networks seem to be suitable for a task like sequence-to-sequence prediction, we did not elaborate on various open questions concerning this topic: Are raw pose estimates the best input for such a network or should we leverage secondary features like limb angles? How can the drawbacks of wide dilated convolutions, namely high volatility on the edge of the output signals, be diminished or prevented? Can the proposed networks be generalized for other sequence-to-sequence tasks? Is it possible to invert the problem, i.e., predict the motion sequence from a series of force measurements? All these questions and many more remain future work.

# List of Figures


2.1 An example of a human pose graph. . . . . 10
2.2 Examples images from the swimming channel dataset. . . . . 17
2.3 Dynamic range of motion for different datasets. . . . . 20
2.4 Velocity vectors of the left body side of a swimmer. . . . . 22
2.5 Examples images from the ski jumper dataset. . . . . 24

3.1 A swimmer in a swimming channel with different poselet arm patches. 28
3.2 A topological visualization of different distance (semi)metric embeddings. . . . . 34
3.3 Poselet similarity examples from the swimmer data set. . . . . 36
3.4 Poselet similarity in the LSP data set. . . . . 37
3.5 All four key-poses evaluated in the chapter. . . . . 44
3.6 PCKF for maximum likelihood, prior and maximum posteriori. . . . . 46
3.7 PCKF for a varying number of poselets and human gait dataset. . . . 47
3.8 Examples from an additional camera view of the swimming channel. . 48
3.9 Examples from the CMU Motion of Body database. . . . . 50

4.1 Image dependent pairwise relations. . . . . 54
4.2 A part detection network based on the AlexNet structure. . . . . 56
4.3 One step of the dynamic program. . . . . 60
4.4 The CG-IRLS graph structure for human pose estimation. . . . . 62
4.5 A convolutional pose machine (CPM) network. . . . . 63
4.6 The CPM VGG network derived from the classical CPM. . . . . 64
4.7 PCK curves of CG-IDPR and CPM networks. . . . . 68
4.8 Two swimmer poses estimated by the CG-IDPR model. . . . . 70

5.1 Qualitative examples of different errors. . . . . 76

5.2 A sequence of joint coordinates of a freestyle swimmer's elbows. . . . 78
5.3 Pearson correlation coefficients for residual magnitudes of different joints. . . . 79
5.4 An example for a kinematic partner graph for wrist joints. . . . 81
5.5 Velocity edges between joint detections of partner joints. . . . 84
5.6 Percentage of Correct Keypoints for all four major swimming styles. . 91
5.7 Example of key-poses defined by human experts. . . . 94

6.1 The different phases of a ski jump. From left to right: in-run, jumping, flight phase, landing. . . . 98
6.2 Based on the predicted poses of an athlete (green/yellow), joint trajectories (blue, orange, dark red) and camera offset $\Delta x$ and $\Delta y$ for two camera views can be inferred with the method described in Section 6.2.2. The red line indicates the edge of the take-off table, which serves as the anchor for force-image synchronization. The synchronized force signal for the jump is depicted as a red graph. . . . 99
6.3 A standard TCN block inspired. . . . 105
6.4 An exemplary network architecture comprised of 6 TCN blocks. . . . 107
6.5 Examples of pose estimates predicted by a fine-tuned CPM-VGG network. . . . 111
6.6 Percentage of correct keypoints for the fine-tuned CPM-VGG for ski jumpers. . . . 111
6.7 Qualitative outputs of different architectures. . . . 113
6.8 Percentage of correctly identified key-frames depicting the moment of the maximal leg pressure. . . . 114

# List of Tables


2.1 A summarization of the *mmc-swim-channel* dataset. . . . . 18

3.1 Scores for the key-pose detector for different probabilistic estimators. 49

4.1 Comparison of PCK scores . . . . . 69
4.2 PCKFs for N-TS, RF-TS and ML-AL. . . . . 71

5.1 Mean residual error (x,y) for orientation normalized wrist and ankle residuals. . . . . 80
5.2 Mean residual error (x,y) for orientation normalized wrist and ankle residuals after correction. . . . . 88
5.3 RMS values of the Euclidean distance between prediction and ground truth for all joints. . . . . 93
5.4 PCKF and Precision of key-pose retrieval. . . . . 95

6.1 Overview over different tested network architectures. . . . . 110
6.2 MSE and median of squared errors between ground truth and predicted forces. . . . . 112

# Bibliography


[1] S. Diamond A. Agrawal, R. Verschueren and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[2] K. Aftab, R. Hartley, and J. Trumpf. Generalized weiszfeld algorithms for lq optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(4):728–745, April 2015.

[3] I. Akhter, T. Simon, S. Khan, I. Matthews, and Y. Sheikh. Bilinear spatiotemporal basis models. *ACM Transactions on Graphics*, 31(2):17:1–17:12, April 2012.

[4] R. Al-Rfou, G. Alain, A. Almahairi, et al. Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688, 2016.

[5] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009. Best Paper Award Honorable Mention by IGD.

[6] A. Aristidou and J. Lasenby. Real-time marker prediction and CoR estimation in optical motion capture. *The Visual Computer*, 29(1):7–26, Jan 2013.

[7] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[8] J. S. Beggs. *Kinematics*. Hemisphere Publishing Corporation, 1983.

[9] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. In *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, pages 468–475, May 2017.

[10] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, Mar 1994.

[11] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *European Conference on Computer Vision (ECCV)*, 2010.

[12] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision (ICCV)*, 2009.

[13] M.A. Brubaker, L. Sigal, and D.J. Fleet. Estimating contact dynamics. In *Proc. IEEE International Conference in Computer Vision (ICCV)*, 2009.

[14] S. Carlsson and J. Sullivan. Action recognition by shape matching to key frames. In *IEEE Computer Society Workshop on Models versus Exemplars in Computer Vision*, 2001.

[15] X. Chen and A. L. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1736–1744. Curran Associates, Inc., 2014.

[16] Y. Chen, C. Shen, X. Wei, L. Liu, and J. Yang. Adversarial posenet: A structure-aware convolutional network for human pose estimation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1221–1230, 2017.

[17] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics, 2014.

[18] W. S. Cleveland. Lowess: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician,*, 35:54, 1981.

[19] R. Cutler and L. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:781–796, 1999.

[20] N. Dalal and .l Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[22] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[23] P. F. de de Dios, Q. Meng, and P. W. H. Chung. A machine learning method for identification of key body poses in cyclic physical exercises. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*, SMC '13, pages 1605–1610, Washington, DC, USA, 2013. IEEE Computer Society.

[24] C. M. de Souza Vicente, E. R. Nascimento, L. E. C. Emery, C. A. G. Flor, T. Vieira, and L. B. Oliveira. High performance moves recognition and sequence segmentation based on key poses filtering. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016.

[25] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[26] M. Einfalt, D. Zecha, and R. Lienhart. Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming. In *2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018*, pages 446–455, 2018.

[27] H. Eng, K. Toh, W. Yau, and J. Wang. Dews: A live visual surveillance system for early drowning detection at pool. *IEEE Trans. Circuits Syst. Video Techn.*, 18(2):196–210, 2008.

[28] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[29] S. R. Fanello, C. Keskin, P. Kohli, S. Izadi, J. Shotton, A. Criminisi, U. Pattacini, and T. Paek. Filter forests for learning data-dependent convolutional kernels. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 1709–1716, June 2014.

[30] H. Fani, A. Mirlohi, H. Hosseini, and R. Herperst. Swim stroke analytic: Front crawl pulling pose classification. In *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10, 2018*, pages 4068–4072, 2018.

[31] M. Fastovets, J. Guillemaut, and A. Hilton. Athlete pose estimation from monocular tv sports footage. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013.

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

[33] 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(9):1627–1645, Sept 2010.

[34] 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(9):1627–1645, Sept 2010.

[35] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, September 2010.

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

[37] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. Comput.*, 22(1):67–92, January 1973.

[38] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. *International Conference on Learned Representations (ICLR)*, 2016.

[39] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 4346–4354, Washington, DC, USA, 2015. IEEE Computer Society.

[40] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017.

[41] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran. Detect-and-track: Efficient pose estimation in videos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 350–359, 2018.

[42] G. Gkioxari, P. Arbelaez, L. Bourdev, and J. Malik. Articulated pose estimation using discriminative armlet classifiers. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:3342–3349, 2013.

[43] Georgia Gkioxari, Alexander Toshev, and Navdeep Jaitly. Chained predictions using convolutional neural networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 728–743. Springer, 2016.

[44] J.C. Gower. Statistical methods of comparing different multivariate analyses of the same data. *Mathematics in the Archeaelogical and Historical Sciences*, pages 138–149, 1971.

[45] P. J. Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. jrstatsocb, 1984.

[46] T. Greif and R. Lienhart. A kinematic model for Bayesian tracking of cyclic human motion. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7543 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, January 2010.

[47] R. Gross and J. Shi. The cmu motion of body (mobo) database. Technical Report CMU-RI-TR-01-18, Robotics Institute, Pittsburgh, PA, June 2001.

[48] L. Haiping, K. Plataniotis, and A. Venetsanopoulos. A layered deformable model for gait analysis. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 249–254, April 2006.

[49] K. Hakozaki, N. Kato, M. Tanabiki, J. Furuyama, Y. Sato, and Y. Aoki. Swimmer's stroke estimation using cnn and multilstm. *Journal of Signal Processing*, 22(4):219–222, 2018.

[50] H. Hatze. The meaning of the term: "biomechanics". In *Journal of Biomechnics 7*, pages 189–190, 1974.

[51] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[52] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1026–1034, Washington, DC, USA, 2015. IEEE Computer Society.

[53] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[54] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[55] S. Hong, H. Lee, and E. Kim. Fusion of multiple gait cycles for human identification. In *ICCAS-SICE, 2009*, pages 3171–3175, Aug 2009.

[56] J. Hwang, S. Park, and N. Kwak. Athlete pose estimation by a global-local network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[57] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015.

[58] U. Iqbal, M. Garbade, and J. Gall. Pose for action - action for pose. *IEEE Conference on Automatic Face and Gesture Recognition*, 2017.

[59] U. Iqbal, A. Milan, and J. Gall. Posetrack: Joint multi-person pose estimation and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[60] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM.

[61] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[62] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[63] E. Jones, T. Oliphant, and et al. P. Peterson. SciPy: Open source scientific tools for Python, 2001–.

[64] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *CoRR*, abs/1610.10099, 2016.

[65] L. Ke, M. Chang, H. Qi, and S. Lyu. Multi-scale structure-aware network for human pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25, NIPS 2012*, pages 1097–1105. Curran Associates, Inc., 2012.

[67] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[68] K. Lee and R.D. Green. Temporally synchronizing image sequences using motion kinematics. In *Proc. Image and Vision Computing*, June 2005.

[69] C. Li, Z. Zhang, W. S. Lee, and G. H. Lee. Convolutional sequence to sequence model for human dynamics. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, abs/1805.00655, 2018.

[70] R. Lienhart, M. Einfalt, and D. Zecha. Mining automatically estimated poses from video recordings of top athletes. *Int. J. Comp. Sci. Sport*, 17(2):94–112, 2018.

[71] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[72] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[73] U. Mall, G. Roshan Lal, S. Chaudhuri, and P. Chaudhuri. A deep recurrent framework for cleaning motion capture data. *CoRR*, abs/1712.03380, 2017.

[74] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, Piscataway, NJ, USA, July 2017. IEEE.

[75] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian image understanding: Unfolding the dynamics of objects in static images. *CoRR*, abs/1511.04048, 2015.

[76] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi. "what happens if..." learning to predict the effect of forces in images. *CoRR*, abs/1603.05600, 2016.

[77] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[78] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, pages 483–499, 2016.

[79] F. Padua, R. Carceroni, G. Santos, and K. Kutulakos. Linear sequence-to-sequence alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):304–320, Feb 2010.

[80] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *International Conference on Computer Vision (ICCV)*, 2015.

[81] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Strong appearance and expressive spatial models for human pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[82] D. Ramanan. Learning to parse images of articulated bodies. In P. B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1129–1136. MIT Press, 2007.

[83] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 2650–2657, Washington, DC, USA, 2013. IEEE Computer Society.

[84] C. X. Ries and R. Lienhart. Automatic pose initialization of swimmers in videos. volume 7543, page 75430J. SPIE, 2010.

[85] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[86] B. Sapp and B. Taskar. MODEC: multimodal decomposable models for human pose estimation. In *CVPR*, pages 3674–3681. IEEE Computer Society, 2013.

[87] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[88] G. A. F. Seber. *Multivariate observations*. Wiley series in probability and mathematical statistics. Wiley, New York, NY [u.a.], 1984.

[89] L. Sha, P. Lucey, S. Morgan, D. Pease, and S. Sridharan. Swimmer localization from a moving camera. In *DICTA*, pages 1–8. IEEE, 2013.

[90] L. Sha, P. Lucey, S. Sridharan, S. Morgan, and D. Pease. Understanding and analyzing a large collection of archived swimming videos. In *IEEE Winter Conference on Applications of Computer Vision, Steamboat Springs, CO, USA, March 24-26, 2014*, pages 674–681, 2014.

[91] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017.

[92] R. Sibson. Studies in the robustness of multidimensional scaling: Procrustes statistics. *Journal of the Royal Statistical Society. Series B (Methodological)*, 40(2):pp. 234–238, 1978.

[93] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[94] J. Song, L. Wang, L. Van Gool, and O. Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[95] W. Tang, P. Yu, and Y. Wu. Deeply learned compositional models for human pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[96] G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1345–1352. MIT Press, 2007.

[97] J. J. Tompson, A. Jain, Y. Lecun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In Z. Ghahramani, M. Welling, C. Cortes, N.d. Lawrence, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1799–1807. Curran Associates, Inc., 2014.

[98] X. Tong, L. Duan, C. Xu, Q. Tian, and H. Lu. Local motion analysis and its application in video based swimming style recognition. In *ICPR (2)*, pages 1258–1261. IEEE Computer Society, 2006.

[99] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[100] A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[101] T. Tsumita, H. Shishido, I. Kitahara, and Y. Kameda. Swimmer position estimation by lane rectification. In *International Workshop on Advanced Image Technology (IWAIT) 2019*, volume 11049. SPIE, 2019.

[102] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[103] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[104] B. Victor, Z. He, S. Morgan, and D. Miniutti. Continuous video to simple signals for swimming stroke detection with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[105] D. Wedge, P. Kovesi, and Du Huynh. Trajectory based video sequence synchronization. In *Digital Image Computing: Techniques and Applications (DICTA'05)*, pages 13–13, Dec 2005.

[106] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[107] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 127–135. Curran Associates, Inc., 2015.

[108] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang. Learning feature pyramids for human pose estimation. In *arXiv preprint arXiv:1708.01101*, 2017.

[109] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *CVPR*, 2016.

[110] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1385–1392. IEEE, 2011.

[111] D. Zecha, C. Eggert, M. Einfalt, S. Brehm, and R. Lienhart. A convolutional sequence to sequence model for multimodal dynamics prediction in ski jumps. In *Proceedings of the 1st International Workshop on Multimedia Content Analysis in Sports*, MMSports'18, pages 11–19, New York, NY, USA, 2018. ACM.

[112] D. Zecha, C. Eggert, and R. Lienhart. Pose estimation for deriving kinematic parameters of competitive swimmers. In *Electronic Imaging 2017. Burlingame, CA, USA.*, 2017.

[113] D. Zecha, M. Einfalt, C. Eggert, and R. Lienhart. Kinematic pose rectification for performance analysis and retrieval in sports. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[114] D. Zecha, M. Einfalt, and R. Lienhart. Refining joint locations for human pose tracking in sports videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[115] D. Zecha, T. Greif, and R. Lienhart. Swimmer detection and pose estimation for continuous stroke-rate determination. In *Proc. SPIE*, volume 8304, pages 830410–830410–13, 2012.

[116] D. Zecha and R. Lienhart. Key-pose prediction in cyclic human motion. In *2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015, Waikoloa, HI, USA, January 5-9, 2015*, pages 86–93, 2015.

[117] H. Zhang, H. Ouyang, S. Liu, X. Qi, X. Shen, R. Yang, and J. Jia. Human pose estimation with spatial contextual information. *CoRR*, abs/1901.01760, 2019.

[118] R. Zhang, C. Vogler, and D. Metaxas. Human gait recognition. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, pages 18–18, June 2004.