Master Thesis

An Analysis of Convolutional Pose Machines – Improvements, Extensions and Applications

Moritz Einfalt

13.06.2017

Universität Augsburg
Fakultät für Angewandte Informatik
Multimedia Computing and
Computer Vision Lab

Reviewer
Prof. Dr. Rainer Lienhart

Second Reviewer
Prof. Dr. Elisabeth André

Supervisor
M.Sc. Dan Zecha

Abstract

Convolutional Pose Machines (CPMs) follow the popularity of convolutional neural networks by applying a multi-stage network architecture to the well studied problem of human pose estimation in images and acchieve state-of-the-art results. In this thesis we focus on improving the CPM framework by extending the network architecture and apply it to a real-world scenario regarding swimming athletes. Our main contributions are extensions to input class label information into a fully convolutional architecture and to modify CPMs for continuous pose estimation on videos. Both variants achieve up to +8% correct body joint detections compared to the default CPM architecture. We additionally address how to use CPMs for joint visibility prediction and pose classification. Our findings are not limited to the swimming context but show the general flexibility of the CPM framework.

Kurzbeschreibung

Convolutional Pose Machines (CPMs) wenden das populäre Konzept von Faltungsnetzen in Form einer mehrstufigen Netzarchitektur auf das häufig betrachtete Problem der Schätzung menschlicher Posen in Bilder an und erreichen damit neue Bestleistungen. In dieser Arbeit konzentrieren wir uns auf die Verbesserung des CPM Systems indem wir die Netzarchitektur erweitern und wenden es auf ein reales Szenario mit Leistungsschwimmern an. Die wesentlichen Beiträge dieser Arbeit sind Erweiterungen um Klassenzugehörigkeiten als Eingabe in reinen Faltungsnetzen zu verarbeiten und die CPMs so zu verändern, dass sie für die kontinuierliche Posenschätzung auf Videos verwendet werden können. Beide Varianten erreichen bis zu +8% an korrekten Gelenkdetektionen im Vergleich zur ursprünglichen CPM Architektur. Zusätzlich beschreiben wir, wie CPMs zur Vorhersage der Sichtbarkeit von Gelenken und zum Klassifizieren von Posen verwendet werden können. Unsere Erkenntnisse sind nicht auf den Kontext von Schwimmern beschränkt, sondern zeigen die allgemeine Flexibilität des CPM Systems.

Contents

1 Introduction

The interest in *Computer Vision* has been steadily increasing over the last years and decades. The need for computers to see – and more importantly to understand what they see – is driven by the sheer amount of visual data that is available today. Digital cameras are present everywhere in a multitude of devices due to the advance of cheap consumer electronics. Combined with the ability to store and share images and videos anytime in online storage or social portals, the availability of image data is nearly unlimited. As an example, over 80 million images and videos are shared every day solely via the social platform Instagram (as of May 2016)[1]. But this flood of data has to be organized and made usable, requiring some form of automation: sorting, classifying or labeling images, recognizing image content or identifying objects and humans – tasks that are subsumed by the field of computer vision. Modern computer vision is closely related to *Machine Learning*. While both are separate fields of research, they are influencing and contributing to each other. Both topics benefit heavily from the advance in high performance computing and GPUs in particular, which led to a revival of *Artificial Neural Networks*. They have been applied to many computer vision tasks with great success and are vital to many state-of-the-art vision approaches that require some notion of learning.

Human Pose Estimation is a long standing task in the domain of computer vision. Estimating the configuration of a persons body parts in images or videos is an exemplary task for structured recognition. There exists a large research community that is constantly striving to develop better and more reliable methods. Human pose estimation on common 2D images however is still regarded a challenging task.

With the introduction of *Convolutional Pose Machines* (CPMs), Wei et al. present a new approach to human pose estimation on the foundation of *Convolutional Neural Networks* [WRKS16]. The CPM framework uses a single network that is trained end-to-end to directly predict the pose configuration given an input image. The CPMs were able to outperform all competing approaches to human pose estimation at the time of release and are considered an important advancement [NYD16, IPA+16]. Based on its notable success, this work is dedicated to a thorough analysis of the CPM framework. Our goal is to gain a better understanding of its operation and to identify important design decisions. We further want to apply CPMs to a specialized scenario of human pose estimation in sports and develop multiple extensions to the CPM framework to incorporate additional context information for increased performance and reliability.

[1] https://instagram-press.com/blog/2016/05/11/a-new-look-for-instagram

Overview

In Chapter 2 we define the task of human pose estimation and give a short overview of how methods for pose estimation have evolved. We then describe the CPM framework and its network architecture in detail, reevaluate it on a popular human pose estimation dataset and discuss the effect of certain implementation details.

In Chapter 3 we describe how occluded body joints are handled and how to extend the CPM to explicitly predict joint visibility.

Chapter 4 presents CPMs in the context of sports. We apply them to recordings taken in a swimming channel to facilitate performance assessment for swimmer athletes.

With the example of different swimming styles, we discuss in Chapter 5 how contextual information that limits the variety of poses can be included into the CPM and how this additional information can be inferred if not available.

Chapter 6 is dedicated to continuous pose estimation. We show how sequential information from subsequent video frames can be utilized to extend the single-image operation of CPMs to videos.

In Chapter 7 we discuss how errors detected in pose estimates and possible correction hints can be reapplied to the CPM by encoding additional belief where certain parts of the body might be located.

Finally, we summarize our results in Chapter 8 and motivate future work.

2 Convolutional Pose Machines for Human Pose Estimation

Convolutional Pose Machines (CPMs) join the recent success of deep convolutional neural networks in computer vision. Among others, they achieve state-of-the-art results on challenging benchmarks for human pose estimation. They constitute another important step towards reliable human pose estimation in 2D images that is applicable in real-world scenarios. At the same time, they introduce simple yet efficient ideas of how convolutional neural networks can be used in this domain. Since CPMs form the foundation of this work, this chapter gives a thorough introduction to the CPM framework.

It is first necessary to define the task of human pose estimation in Section 2.1 and describe what challenges it poses.

Section 2.2 gives a brief overview of the development of human pose estimation, up to the recent integration of convolutional neural networks.

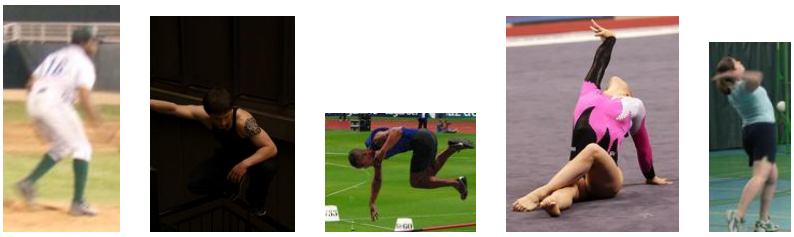
Section 2.3 presents the CPM framework in detail: Its main ideas and innovations, the architecture itself, a reevaluation on a popular human pose estimation benchmark and an analysis of certain implementation details.

2.1 Human Pose Estimation in 2D Images

For this work and in related literature, the task of human pose estimation is defined as follows: given an 2D color image of a person, locate and identify the body parts or joints of this person. In the case of the CPM framework, the task is further constrained. It is known that the image does indeed depict a person and its rough location is given. Thus, the detection of the person as a whole is not part of the task. This is common for recent publications and the relevant benchmarks, but both tasks are also often considered simultaneously (see e.g. [ARS09]).

Possible applications of human pose estimation are manifold. They include movement-based controlling and interaction, surveillance and tracking, activity recognition or diagnosis in medicine and sports, a domain that will be further investigated in Chapter 4.

The difficulty of human pose estimation on 2D images stems from different aspects of the problem (see Figure 2.1 for examples). First of all, it shares the same challenges that are common to many problems in computer vision. This includes image



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

Figure 2.1: Examples from the Leeds Sports Pose (LSP) dataset that show some of the main challenges in human pose estimation: (a) varying image sharpness and motion blur, (b) challenging lighting conditions, (c) variation in orientation, (d) foreshortening (legs), (e) self-occlusion (left arm).

variation in sharpness or resolution and challenging lighting conditions. Humans as the object of interest in images can vary with respect to scale and rotation. Their visual appearance depends on multiple factors, mainly the size and shape of the body, the skin color and the clothing. The appearance of the background is nearly unlimited.

The actual task requires to identify the pose of a person. The human pose is defined by the configuration of all parts of the body. The space of possible poses is large: the joints of the human body allow a large degree of freedom in how body parts can be located and oriented relative to each other. This is also the reason why the task is often referred to as *Articulated Pose Estimation*. At the same time, the anatomy of humans clearly constrains their pose, e.g. an arm is always connected to the torso at the shoulder. For a pose estimation system it is necessary to infer information about the human pose as a whole in order to reliably identify a body part. For example, the lower arms of a person can look very similar. To identify left and right limbs, additional context like the orientation of the torso or head is required.

At last, the information available to the human pose estimation system is only a 2D projection of the pose configuration in 3D space. The rotation of body parts outside of the projection plane can lead to foreshortening. And occlusion, either by different body parts (self-occlusion) or other objects in the scene can lead to situations where limbs are completely or partially hidden.

All these difficulties combined make the design of a precise and reliable human pose estimation system a challenging task. But with the rediscovery of convolutional neural networks, multiple promising methods are available today.

2.2 Convolutional Neural Networks for Human Pose Estimation

The traditional approach to human pose estimation are *Pictorial Structures*, originally developed for object detection in general [FH05]. The object of interest is modeled by a collection of parts that are connected to each other. The connections between parts are deformable, e.g. the distance between connected parts or their relative orientation can vary. This design is naturally applicable to human pose estimation, where we are explicitly interested in the location of individual body parts with a fixed connection pattern. Pictorial Structures use a probabilistic framework. They separately define the appearance of individual parts and how the configurations, i.e. location, scale and rotation of connected parts depend on each other. These dependencies can be represented in a graph-based model with nodes for each part and edges for all pairs of connected parts. The overall goal is to find the most probable part configuration given the input image. Efficiently inferring the necessary probabilities is possible if certain conditions are met. Most prominently, the model is limited to tree graphs, i.e. no cyclic dependencies between parts are allowed.

The subsequent literature on pictorial structures is vast and far beyond the scope of this work. It focuses on the shortcomings and limiting requirements of the original approach, e.g. better part detectors using rich image features [ARS09] or more complex part interactions [LH05, ST13]. A survey of pictorial structures and related model-based approaches for human pose estimation can be found in [PSEAG14].

With the recent success of deep convolutional neural networks for image classification [KSH12], this concept is also applied to further topics in computer vision like object detection [GDDM14] and image segmentation [LSD15]. The *DeepPose* architecture is among the first approaches to human pose estimation solely based on deep convolutional neural networks [TS14]. One important benefit compared to pictorial structures is that the manual design of body part interactions or part detectors based on hand-designed image features is not required. The network is intended to learn these mechanisms itself. The same motivation is present in Convolutional Pose Machines, proposed in [WRKS16]. Using a deep, multi-stage convolutional neural network, they are able to outperform many of the best human pose estimation methods for 2D images. We give a detailed description of this approach in the remainder of this chapter.

2.3 The Convolutional Pose Machine Framework

The CPM framework is developed for pose estimation on 2D RBG images of humans using a deep convolutional neural network. The learning objective is to estimate

the location of major joints of the human body that define its pose. In the default setting of the CPM, these are $J = 14$ joints consisting of the top of the head, the neck, the shoulders, elbows and wrists (left and right) and the hip, knees and ankles (left and right). Figure 2.2a depicts an example for this joint-based pose definition.

The main motivation for using a deep convolutional neural network is to implicitly learn to utilize spatial context. That is, not detecting individual joints separately but to learn dependencies between any set of joints. This is necessary for improved estimates on difficult instances, e.g. with uncommon poses or occluded parts of the body. The ability to reason about the entire pose helps to overcome the limited part interactions of pictorial structures.

The framework uses an iterative network architecture that repeatedly refines an initial pose estimate. The idea of repeated estimation and refinement can be found in multiple architectures for human pose estimation [TS14, PCZ15]. In fact, many of the ideas incorporated in Convolutional Pose Machines stem from the *Pose Machine* framework [RMH+14], which can be seen as their predecessor. It uses a similar stage-wise structure, but detects joints with random forests trained on HOG image features. These are replaced by a single convolutional neural network in the CPM framework which leads to an overall much simpler design.

2.3.1 Network Architecture

The CPM uses a pure convolutional neural network, divided into S stages. The network is trained on instances $(x, \mathbf{y})$, where x is the input RGB image of fixed size, centered on the person of interest. [WRKS16] use an image size of 368×368 for their best results. $\mathbf{y} = (y_1, \dots y_J)$ is the ground truth location of all J joints in cartesian image coordinates. The objective of the network is to regress confidence values for all possible joint locations. The output of every stage s is a stack of confidence maps (in the following simply denoted heatmaps) $\hat{\mathbf{h}}^s = (\hat{h}^s_1, \dots, \hat{h}^s_{J+1})$. For every location $z \in \mathcal{Z}$, $\hat{h}^s_j(z)$ is interpreted as the confidence that joint j is located at z. The set of possible location $\mathcal{Z}$ is

$$\mathcal{Z} = \left\{ (u, v) \,|0 \leq u < \left\lceil \frac{w}{8} \right\rceil, 0 \leq v < \left\lceil \frac{h}{8} \right\rceil \right\}, \tag{2.1}$$

with an input image of size $w \times h$. Hence, the network produces confidence values for every 8×8 patch in the input image. The last heatmap $J + 1$ subsumes the background and images regions without a body joint: it is trained to have high confidence when no joint is located at the respective position.

Figure 2.3 depicts the stage-wise network architecture, which we now gradually describe. Each stage resembles a classical convolutional neural network for image classification. It performs repeated convolution and max pooling on the quadratic input image. Every convolution layer except the last one in each stage is followed

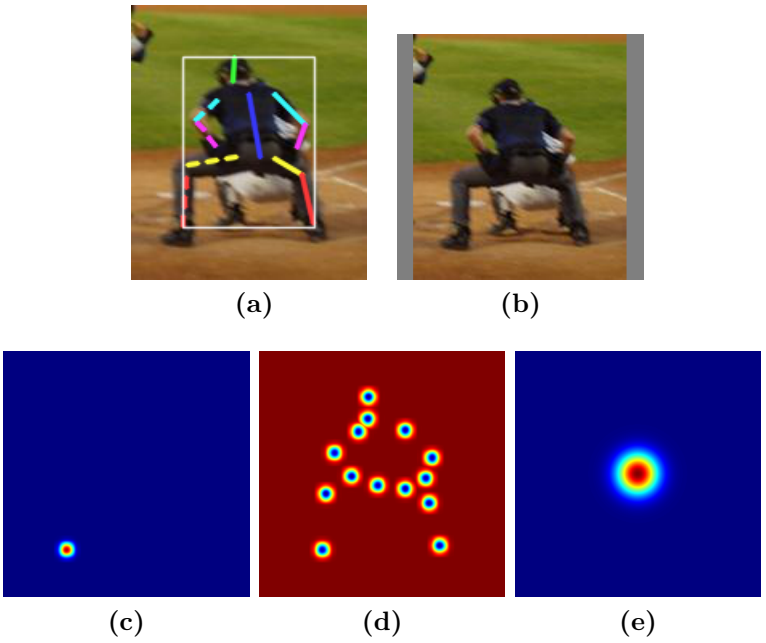


Figure 2.2: CPM network input and ground truth encoding. (a) Example image with annotated joint locations and surrounding bounding box. The human pose is visualized by drawing connections between adjacent joints. Limbs on the left side of the body are stroked. (b) The CPM input image is scaled, cropped and padded to a fixed quadratic size. (c) 2D heatmap encoding the ground truth location of the left ankle. (d) Ground truth for the background. (e) Additional network input encoding the center of the person.

by a rectified linear unit (ReLU). After three pooling layers, each stage operates on a 8-times downscaled input size of 46×46. This equals the size of the output heatmaps.

2.3.1.1 Estimation with Local Image Evidence

The first CPM stage $s = 1$ is intended to predict joint locations using only local image evidence. This follows from the limited *receptive field* of this stage: each location in the output heatmaps of layer `conv7` is only connected to a surrounding 160×160 patch of the input image. The decision whether a joint is located at a specific position is thus limited to local image information. The first network stage can therefore only reason about direct visual appearance of joints and their local connectivity pattern to adjacent joints. It can not reason about the complete input image and thus the whole pose. This makes it difficult to distinguish joints that share similar appearance but are located far apart. The resulting network output thus often contains activations at multiple locations in a joint heatmap (see Figure 2.4a). The heatmap encoding is crucial to allow the network to express

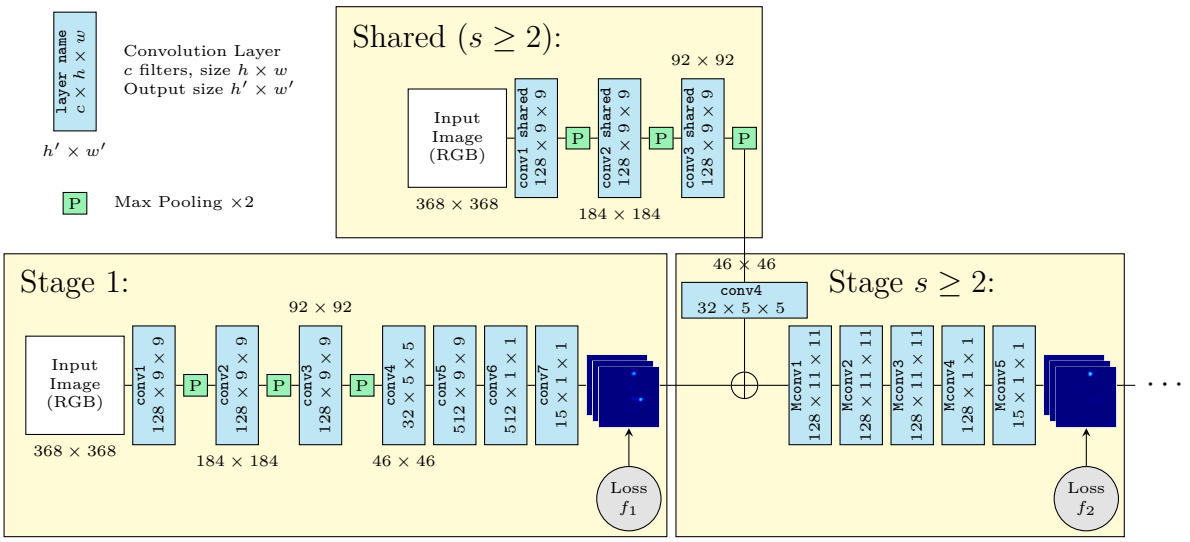


Figure 2.3: CPM network architecture for a pose configuration with $J = 14$ joints and an input image size of 368×368. Each stage produces refined estimates for joint locations using the input image and the estimate from the previous stage. Shared layers are only instantiated once and used by all stages $s \geq 2$.

multiple hypotheses (i.e. possible joint locations). It is a main difference to the DeepPose architecture that directly regresses the cartesian image coordinates for each joint [TS14].

2.3.1.2 Refinement with Larger Spatial Context

All subsequent stages $s \geq 2$ in the CPM share the same structure. They operate on the input image and the joint heatmaps from the preceding stage and output a set of refined heatmaps. The layers `conv1-3` for image features are shared between all stages $s \geq 2$, the remaining layers `conv4` and `Mconv1-5` are instantiated for each stage separately. Beginning with layer `Mconv1`, image features and estimated heatmaps from the preceding stage are concatenated and processed jointly. Due to the additional spatial convolution layers, the receptive field of the network is gradually increasing. After stage 2, the receptive field is already of size of 400×400 and thus covers large parts of the input image from stage 1. The additional spatial context enables the network to learn the relationship between joints (or body parts in general) to resolve ambiguities in the estimate of previous stages. This includes for example the differentiation of the left and right ankle or other joints with similar appearance (see Figure 2.4c).

The total number of stages S is variable, since each stage acts as an additional building block for further pose refinement.

(a) (b) (c)

Figure 2.4: CPM estimate of the left ankle after stages 1 - 3. Output heatmaps are superimposed on the input image for easier interpretation. (a) Limited receptive field (white rectangle) of stage 1 leads to activations on both ankles. (b), (c) Additional spatial context in subsequent stages helps to uniquely identify the left ankle.

2.3.1.3 Efficient Learning with a Stage-wise Loss Definition

To enable learning, the predicted pose has to be evaluated. For this purpose, the ground truth for joint j is represented using an artificial heatmap h_j with a fixed-sized gaussian activation at location y_j. The gaussian is normalized such that its maximum is 1. The ground truth heatmap h_{J+1} for the background is obtained by taking the location-wise maximum over all joint heatmaps h_j with $j \leq J$ and inverting it:

$$h_{J+1}(z) = 1 - \max_j h_j(z) \; \forall z \in \mathcal{Z}. \tag{2.2}$$

The collection of all ground truth heatmaps is denoted $\mathbf{h} = (h_1, \ldots h_{J+1})$. A visualization of the the ground truth encoding can be seen in Figure 2.2.

The network is trained using stochastic gradient descent (SGD). Training loss is calculated for each stage by comparing the predicted and ground truth heatmaps. For each training example $(x, \mathbf{y})$, the loss f_s after stage s is defined as an euclidean loss with

$$f_s(x, \mathbf{y}) = \sum_{j=1}^{J+1} \sum_{z \in \mathcal{Z}} \left\| \hat{h}_j^s(z) - h_j(z) \right\|_2^2. \tag{2.3}$$

Note that neither x nor $\mathbf{y}$ directly appear in the loss definition. However, every $\hat{h}_j^s$ is a function of x and every ground truth heatmap h_j is a function of $\mathbf{y}$. To avoid confusion, f_s is from now on used without the explicit arguments.

[WRKS16] argue that separate loss calculation after every stage – which they denote *intermediate supervision* – is crucial for efficient learning. A common issue in training deep neural networks with SGD is the effect of *vanishing gradients*: after

the loss at the last layer of the network is computed, the gradient with respect to the network parameters decreases in magnitude during backpropagation. This hinders or inhibits learning the parameters of the first layers in the network. The same effect can be observed in the CPM network, if the loss is only computed for the last stages output. By adding the loss terms f_s for all preceding stages, backpropagation refreshes the gradient after every stage such that all network layers show learning progress. This shows another implicit advantage of the stage-wise architecture: intermediate network representations can be interpreted and participate in the loss definition.

2.3.2 Training

2.3.2.1 Data

In [WRKS16], the CPM is trained and evaluated on multiple human pose estimation datasets: MPII Human Pose [APGS14], FLIC [ST13] and the Leeds Sports Pose (LSP) dataset [JE10]. In this work, only the latter is used and thus described in detail. The original LSP dataset contains 2k images of humans during sport activities. Nearly all images depict the whole person without truncation of any body parts. The pose is annotated using the 14 joints configuration described earlier. Joint annotations are person-centric, i.e. joints like the left ankle are defined from the persons point of view. The usual partition is 1k images for training and 1k images for evaluation.

There exists an extension of the LSP dataset (LSPe) that adds another 10k images for training with more variety in poses [JE11]. This leads to a total number of 11k training images. Examples from LSP and LSPe can be seen in Figure 2.1.

For CPM training, the data is further augmented. In each iteration, training images are scaled, rotated and left-right flipped randomly. The scale is drawn uniformly from $[0.7, 1.3]$, the rotation angle uniformly from $[-40°, 40°]$. The probability to flip the image is 0.5. This is done separately for each image. The joint annotations are transformed equally before the ground truth heatmaps are generated. The augmentation leads to a much higher variety in size and orientation and is intended to increase scale and rotational invariance.

2.3.2.2 Learning Parameters

The network is trained in an end-to-end fashion. [WRKS16] show that jointly training all network stages results in superior performance and faster learning compared to a separate training of each stage. All network weights are initialized randomly according to a tight gaussian distribution with zero mean. The initial learning rate has to be set to $8 \cdot 10^{-5}$ to ensure converging network weights. This is notably

lower compared to traditional deep convolutional neural networks [SZ15]. To ensure stable gradients, weight updates are performed batch-wise with a batch size of 16 images.

For optimal results, a 6-stage CPM on LSP is trained for $395k$ iterations, with one batch per iteration. The learning rate is gradually reduced by $\frac{1}{3}$ after every $100k$ iterations. The results on LSP in [WRKS16] indicate only a marginal difference to a CPM with only 3 stages. For this reason, 3-stage CPMs are used throughout this work to keep training and evaluation on the available hardware feasible. The same learning parameters as for the 6-stage version are used. Training the 3-stage CPM on LSP takes 6 days using a pair of NVIDIA GTX 1080 GPUs.

2.3.3 Implementation Details

The CPM network is implemented with the *caffe* library for deep artificial neural networks [JSD+14]. [WRKS16] provide additional tools and demo code in MATLAB to facilitate CPM training and evaluation. For this work, we develop a convenient interface in Python to easily train, evaluate and deploy CPMs.

The CPM implementation contains several important details that are not or only briefly mentioned in the publication.

2.3.3.1 Bounding Box Estimation

The CPM network is trained on square images centered around the person with some padding on all four sides. Any raw input image therefore has to be scaled, cropped and padded such that it meets these requirements. Person detection in a larger image is not part of the CPM. However, LSP does not provided explicit annotations for the location and center of the person or its size. [WRKS16] use the joint annotations to form a tight surrounding bounding box (see Figure 2.2a. Center and size of this box are then used to estimate the center and scale of the person. This has to be done during training and evaluation. Using joint annotations to estimate the necessary properties is of course not applicable to real-world human pose estimation on raw input images. Section 4.3 considers a solution for such a scenario.

2.3.3.2 Scale Search

To obtain the reported pose estimation results in [WRKS16], an explicit scale search is performed. Each test set image is processed 7 times by the CPM with varying scale. Scaling values are in $[0.7, 1.3]$. The resulting output heatmaps from each forward pass are then scaled back to the original image size and averaged. This is done to ensure scale invariance and enable more precise joint localization.

2.3.3.3 Additional Center Map

[WRKS16] use an additional input channel to encode the center of the person. It is a single map with the 8-times pooled size of the input image. The same gaussian encoding as for the ground truth heatmaps is used, but with a much larger standard deviation (see Figure 2.2e). This center map is added to the concatenation preceding layer `Mconv1` in every stage $s \geq 2$. Since data augmentation does not include horizontal or vertical translation, training images are always centered around the person. Thus, the center of the person is implicitly known to be the center of the input image. The additional center map does therefore not provide any additional information. It is simply a constant input throughout training and evaluation. [WRKS16] report that the addition of the center map can increase performance, but do not motivate it any further. The effect of this center map is further discussed in Section 2.3.4.3.

2.3.4 Evaluation

During evaluation, the CMP network produces output heatmaps after every stage. Since later stages are intended to provide better and more refined estimates, only the output of the last stage $s = S$ is used to derive the final prediction of joint locations. The heatmaps are interpreted as confidence maps. The location with highest confidence in each heatmap is therefore used as the predicted location $\hat{y}_j$ of the corresponding joint:

$$\hat{y}_j = \arg\max_{z \in \mathcal{Z}} \hat{h}_j^S(z) . \tag{2.4}$$

If there are multiple locations with maximal confidence, one is chosen randomly. We observe however that this case rarely occurs. Note that before the arg max in Equation 2.4 is applied, the heatmaps are upscaled to the original input image size using bicubic interpolation. Combined with the scale search and heatmap averaging described above, this enables higher precision than the 8-times pooled heatmaps would normally allow. To avoid clutter in the notation, this rescaling step is not explicitly included into the notation.

2.3.4.1 Metric

There exist multiple metrics for both body part and joint based pose estimation. For evaluation on LSP, [WRKS16] use the *Percentage of Correct Keypoints* (PCK) metric. The PCK metric counts a joint as correctly localized if the euclidean distance to the ground truth location does not exceed a fixed fraction α of a reference length. In the original proposal of PCK, the reference length is the maximum of width and height of the bounding box around the person [YR13]. [ST13] instead

use the diameter of the torso, defined as the distance between the left side of the hip and the right shoulder. This is the now common variant of the PCK metric and coincides with the *Percent of Detected Joints* (PDJ) metric [TS14].

With the PCK metric, the prediction $\hat{y}_j$ for joint j is thus correct if

$$\|\hat{y}_j - y_j\|_2 \quad \leq \quad \alpha \cdot \|y_{\text{lhip}} - y_{\text{rsho}}\|_2 \,. \tag{2.5}$$

Common values for α lie within $[0.1, 0.5]$. $\alpha = 0.2$ is often used to compare different human pose estimation systems with a single score [Ins17]. We refer to this specific metric as PCK@0.2.

2.3.4.2 Results on LSP

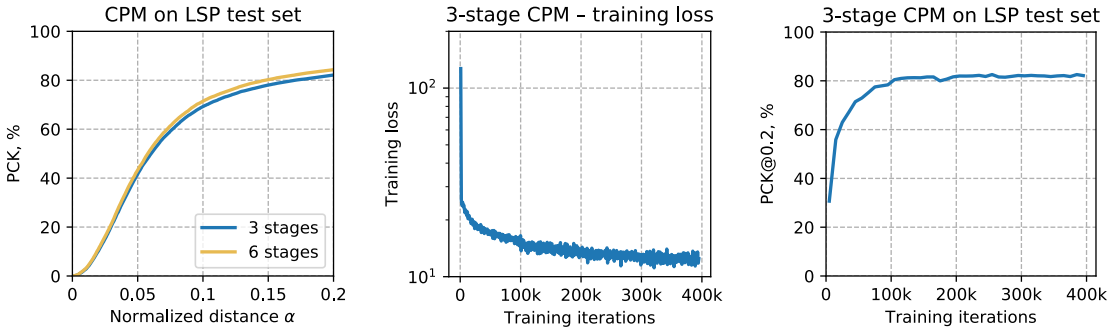
Figure 2.5a depicts the result of the 3-stage CPM on the LSP test set. It shows the PCK scores for $\alpha \in [0, 0.2]$. For comparison, the result of the 6-stage CPM model from [WRKS16] is shown, too. With the fixed PCK@0.2 metric, the 3-stage CPM localizes 82.1% of all joints correctly, the 6-stage version 84.3%. The difference of 2.2% coincides with the results in [WRKS16]. This ensures that the rewritten CPM interface works as intended.

Figure 2.5b and Figure 2.5c depict the training loss and the test set performance as training proceeds. The reduction in loss is concentrated on the first 100k iterations, but it keeps decreasing gradually. Test set performance (PCK) already converges after 200k iterations, but further training does not impose overfitting. Thus, the 3-stage model requires less training time compared to the optimal 395k iterations for the 6-stage model. This is a further motivation to limit all subsequent experiments to a 3-stage model, despite the non-optimal performance.

We visualize qualitative examples of estimated poses on LSP in Figure 2.6. The first row shows successful estimates. They include standard and unusually articulated poses. The second row contains failure cases. Frequent errors are left-right confusion (first image from the right) or double counting of joints due to occlusion (first image from the left).

2.3.4.3 Effect of Center Map

The additional center map input has no real motivation in the original CPM publication. In general, convolution filters are invariant w.r.t. the position they are applied to. That is, the filter performs the same operation when sliding over the layer input. With the explicit center map, it is possible that the network learns filters that only activate on specific regions of the input, e.g. the center. But it is unclear if this facilitates pose estimation on input images where the person is not perfectly centered, especially since such examples never occur during training.



(a) (b) (c)

Figure 2.5: Quantitative results of the 3-stage CPM on LSP. (a) Test set PCK for varying distance thresholds, compared to a 6-stage CPM. (b) Averaged training loss after every 1k iterations (logarithmic scale). (c) Test set PCK@0.2 after every 10k training iterations.

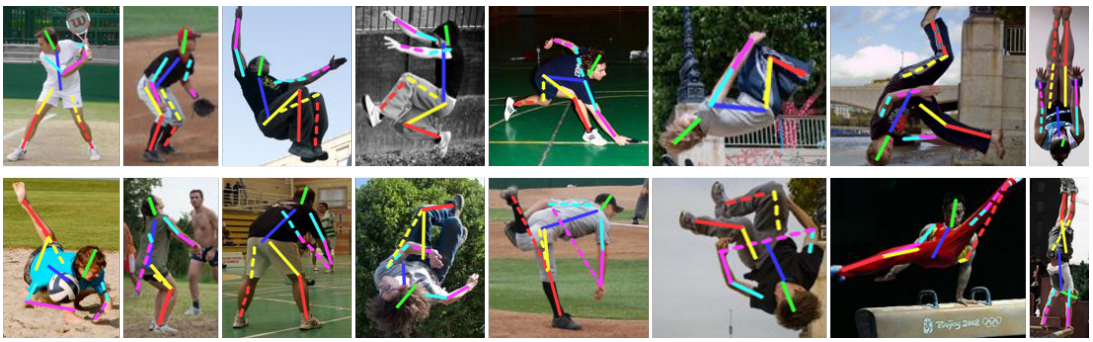


Figure 2.6: Qualitative results of the 3-stage CPM on the LSP test set. First row: pose correctly estimated. Second row: partial failure cases.

Does the network still learn to use the center information as an anchor to locate surrounding joints?

For this purpose, the evaluation on the LSP test set is repeated where input images are translated by a fixed amount. The center map is now either kept as usual (encoding a wrong center), translated equally (encoding the now correct center) or simply set to zero (encoding no center at all). Figure 2.7a illustrates the change in performance when translation is gradually increased. When no shift in x direction is applied, the zeroed center map performs inferior. This is to be expected, since the network expects the constant gaussian map as input. When the input images are shifted, performance in general decreases significantly. First, the networks is not trained on such cases, and second, shifting the images can lead to body parts being truncated an no longer inside the image bounds. When the center map encodes the wrong center, performance is lowest. But interestingly, providing the correct center by shifting the center map by the same amount as the input images does

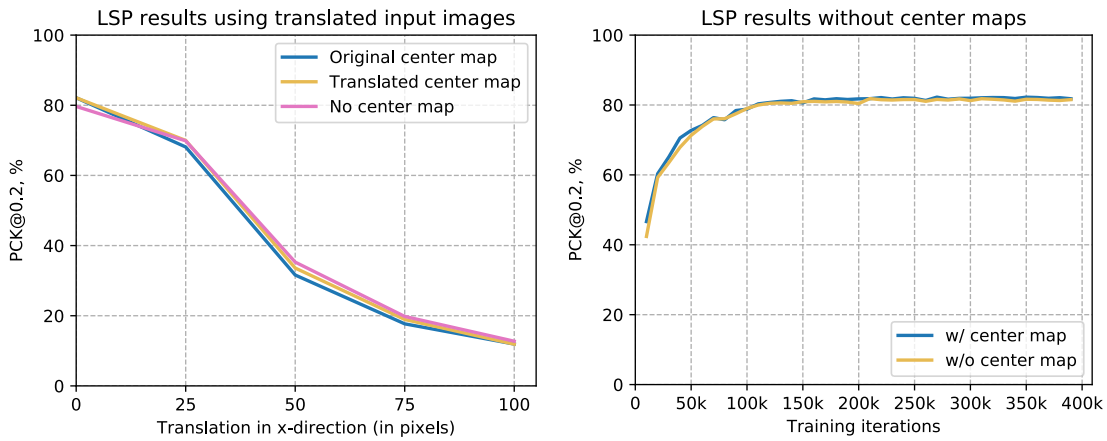


Figure 2.7: Effect of the center map on CPM performance. (a) LSP test set result when input images for the 3-stage CPM are translated by a constant amount. (b) Result when the CPM is trained and evaluated without the center map input.

also not perform best. Simply zeroing the center map leads to the best results here. This shows that the network has not learned to use the center map as an anchoring mechanism to cope with shifted input images.

The question remains if the center map has any benefit for the CPM at all, or if the network has simply learned to cope with this additional input without any semantical meaning. Therefore, the 3-stage CPM is trained a second time from scratch, but without the center map input. We show the LSP test set result after every 10k training iterations in Figure 2.7b, compared to the previous CPM with center map. It can be seen that the version with center map is consistently 0.2%–1% better than the version without. The addition of the center map is indeed beneficial for subsequent layers, but the reason stays unclear. Due to the performance increase, the center map is used in all further experiments.

3 Explicit Prediction of Joint Visibility

One of the main characteristics of the CPM network architecture is its large receptive field to enable the network to learn the geometry of body parts and thus dependencies between joints [WRKS16]. If there is a strong and unambiguous visual clue for a joint, local image information alone can suffice to uniquely locate the joint. This is common for joints with a distinct appearance like the head (see Figure 3.1a). Joints like the wrists or ankles have similar appearance and thus lead to ambiguous visual evidence (see Figure 3.1b). The location of other joints can help to resolve it. But when joints are occluded by other parts of the body (see Figure 3.1c), the network is forced to estimate the location solely based on the geometry of other joints. Thus, the network has to decide implicitly that it can not rely on visual clues because the joint in question is not visible. This motivates the analysis whether this decision – if a joint is visible or not – can be made explicit.

Aside from the scientific nature of this question, there are also practical applications where information about joint visibility can be of use. The task of visibility prediction can be seen as a small step towards 3D pose estimation, because binary depth information of joints has to be inferred. An exemplary application operating on pose and depth information is the Microsoft Kinect, which allows a user to control games and other applications with their body movement [SSK+13]. It is crucial to decide which parts of the body are actually visible to the system, since the user would only expect those to be interpreted as input.

3.1 Baseline Approach

Before discussing how the CPM framework can be extended to provide explicit joint visibility information, the existing work is used to state a baseline approach.

So far, the CPM treats visible and occluded joints equally. During training, the same ground truth representation is used for both kinds of joints: a heatmap with a gaussian peak at the true joint location, normalized to a maximum value of 1. The only exception are training examples with truncated body parts, i.e. joints outside of the image bounds. The ground truth heatmap for these joints is simply empty. During evaluation, the location $\hat{y}_j$ of each joint j is determined by the

(a) Head **(b)** Right ankle **(c)** Right wrist

Figure 3.1: Network output after stage 1, based on local image evidence. (a) Distinct visual evidence leads to correct localization of the head. (b) Ambiguous visual evidence for the right ankle. (c) Right wrist is occluded. Only small network activations on the background.

location of the maximum score in each output heatmap $\hat{h}_j^S$ in the last stage $s = S$ (see Equation 2.4).

In the demo code provided alongside [WRKS16], the scores in the network output are interpreted as a measure of confidence. The lower the maximum score in each heatmap, the lower the confidence in the estimated joint location. All joints with a score lower than a fixed threshold are ignored for the final pose visualization. This is done to suppress possible erroneous joints. Figure 3.2 depicts the visualization of the estimated pose on the example from Figure 3.1. The right wrist and elbow are occluded by the torso, the neck is only partially visible. All three joints have a confidence below the threshold and are thus not shown. The question arises whether occluded or truncated joints can be identified by their confidence value. Is there an incentive that non-visible joints should always lead to lower scores in the network output? This is the case for truncated joints, since the network is trained to produce an empty heatmap that is zero everywhere. Occluded joints however are treated like any visible joint. The network is trained to localize them in the same way as any other joint. Thus, one can not expect lower output scores for such joints in general.

The CPM used in this work is pretrained on the first half of the original Leeds Sports Pose (LSP) dataset and the additional data from LSPe. Both datasets provide visibility information for each joint, i.e. a label $l_j \in \{\text{visible}, \text{occluded}\}$. Table 3.1 contains the number of training examples in the different LSP data sets and the number of joints that are marked as visible, occluded or non-annotated. While all joints in the original LSP data set are annotated, LSPe provides annotations only for visible joints. The ground truth location for occluded joints is not available. Thus, occluded joints in LSPe training examples are represented using an empty ground truth heatmap during CPM training, the same as for truncated joints. LSPe contains ten times as many training examples as the original LSP data set. Therefore, only $\frac{1}{11}$ of all training examples encourage the network to detect occluded

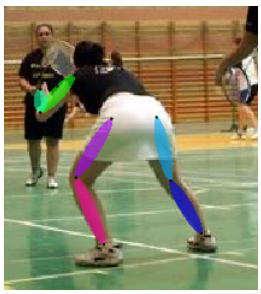


Figure 3.2: Pose visualization from the CPM demo code. Several (partially) occluded joints are suppressed based on the score (confidence) in the network output.

	Images	Visible joints	Occluded joints	Missing annotations
LSP training set	1000	13009	990	1
LSP test set	1000	12942	1054	4
LSPe	10000	123184	0	16816

Table 3.1: Number of visible, occluded and non-annotated joints in the different LSP datasets. LSPe does not contain annotations for occluded joints.

joints with a score of one, while the remaining $\frac{10}{11}$ of training examples enforce an output heatmap that is empty. In combination, this could cause the network to produce scores close to zero for any occluded joint. This motivates a baseline approach to identify occluded joints by low heatmap scores.

3.1.1 Defining an Evaluation Metric for Joint Visibility

In order to measure how well occlusion detection works, it is necessary to specify an evaluation method. The overall problem consists of two tasks: locating and identifying a joint in the image and predicting its visibility. Only if the joint is correctly located in the first place, a prediction about its visibility is meaningful. If for example the CPM misplaced the right wrist in Figure 3.2 onto the location of the left wrist, the reasoning about the right wrists visibility would be based on a wrong region of the image. Wether the joint at this wrong location is visible or not does not relate to the visibility of the right wrists at its actual location. In this chapter, we want to evaluate joint localization and visibility prediction separately. We therefore explicitly limit the evaluation of visibility prediction to joints of the test set that are located successfully. The objective can be specified as a binary classification problem: For each joint in the test set that is located correctly w.r.t. the PCK@0.2 metric, return whether it is visible or occluded.

We define visible joints as positive and occluded ones as negative examples. Any classification partitions the correctly located test set joints into *true positives* (TP), *false positives* (FP), *true negatives* (TN) and *false negatives* (FN). Since binary

classification problems are common in machine learning and beyond, there exist different metrics based on this definition. A popular choice is the *Receiver Operator Characteristic* (ROC), which consists of the fraction of positive examples that are classified correctly (*True Positive Rate*, TPR) and the fraction of negative examples that are misclassified (*False Positive Rate*, FPR) [DG06]:

$$\begin{aligned} \text{TPR} &:= \frac{\text{TP}}{\text{TP} + \text{FN}} \in [0, 1] \\ \text{FPR} &:= \frac{\text{FP}}{\text{FP} + \text{TN}} \in [0, 1] \end{aligned} \tag{3.1}$$

Since each value on its own is meaningless, both TPR and FPR have to be considered simultaneously. The goal is to achieve high TPR while maintaining low FPR.

3.1.2 Baseline Performance

Our baseline approach to visibility prediction uses the 3-stage CPM trained on LSP and LSPe. We will refer to this network as the default CPM throughout this chapter. The procedure for visibility prediction is defined as follows:

1. Process each instance from the test set using the default CPM.
2. For each joint j, extract the location $\hat{y}_j$ and maximum score $\hat{h}_j^S(\hat{y}_j)$ from the corresponding output heatmap after the last stage $s = S$.
3. If the maximum score is lower than a fixed threshold c, label the joint as occluded. Otherwise label it as visible.

Different choices for the threshold c lead to different classification results. The common procedure is to evaluate all possible values for c that lead to distinct classification partitions and thus FPR and TPR values. This can be achieved by collecting the observed joints scores $\hat{h}_j^S(\hat{y}_j)$ for all joints and test set examples. Each score represents one possible choice for c. By combining all resulting pairs of TPR and FPR into a single plot, we obtain the ROC curve in Figure 3.3. Multiple ROC curves can be compared by the area under the curve or the TPR for a fixed FPR value. For a FPR of 0.20 (80% of occluded joints are labeled correctly), a TPR of 0.75 is achieved (75% of visible joints are labeled correctly). This serves as a baseline for further experiments.

3.2 High-Level Network Features for Visibility Prediction

The baseline approach to visibility prediction used the original CPM architecture. The fact that occluded joints are not annotated in the majority of training examples

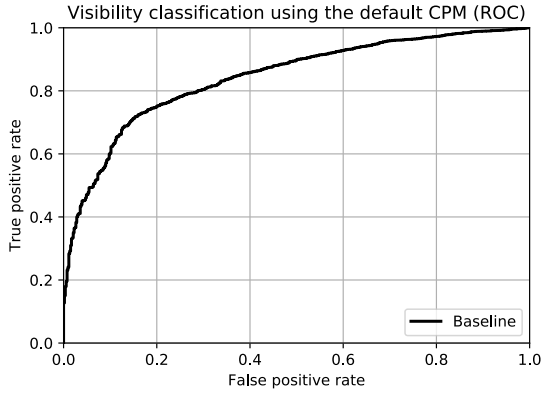


Figure 3.3: Baseline result for visibility prediction on the LSP test set. Joint visibility is decided by reusing the confidence scores in the output heatmaps of the default 3-stage CPM trained on LSP + LSPe.

is exploited to identify occluded joints by low scores in the network output. The task to explicitly classify joints with respect to their visibility is however not part of the learning objective of the default CPM network. How the CPM architecture can be extended to explicitly incorporate this task is presented next.

3.2.1 The Learning Objective

The objective of the original CPM architecture is to locate and identify joints. This is modeled as a regression task, since the network learns to produce an appropriate confidence score at each position in the output heatmaps. In contrast, the new task of visibility prediction is a binary classification task. The simplest idea is to train the network to additionally produce a single label (or class probability) for each joint. This would be appropriate if the existing network produced exactly one location per joint (as e.g. in [TS14]). Due to the heatmap encoding, the network is able to express a multimodal belief about joint locations, as can be seen in Figure 3.1b. For this example, a single label regarding the joint visibility would be ambiguous. It would be unclear to which of the hypothesized joint locations it belonged. This motivates a multimodal design of the joint classification similar to the joint localization task.

For this purpose, the heatmap encoding is adopted to additionally express the joint visibility at each corresponding location. The visibility ground truth v_j for each joint j is defined as a second heatmap with a gaussian peak at the true location y_j of the joint, normalized to a maximum of 1. If the joint is occluded, the gaussian is inverted such that it has value -1 at the ground truth location. The sign of the gaussian peak therefore encodes the class label.

When the network is trained to generate such additional visibility heatmaps $\hat{v}_j$, evaluation works as follows (for each joint j):

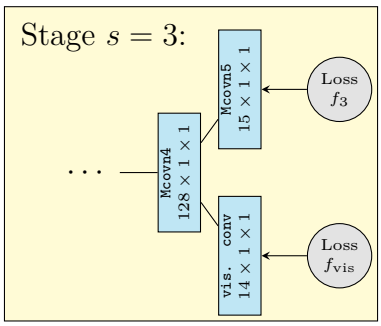


Figure 3.4: Extended CPM for additional joint visibility prediction. Only the last layers of stage 3 are shown.

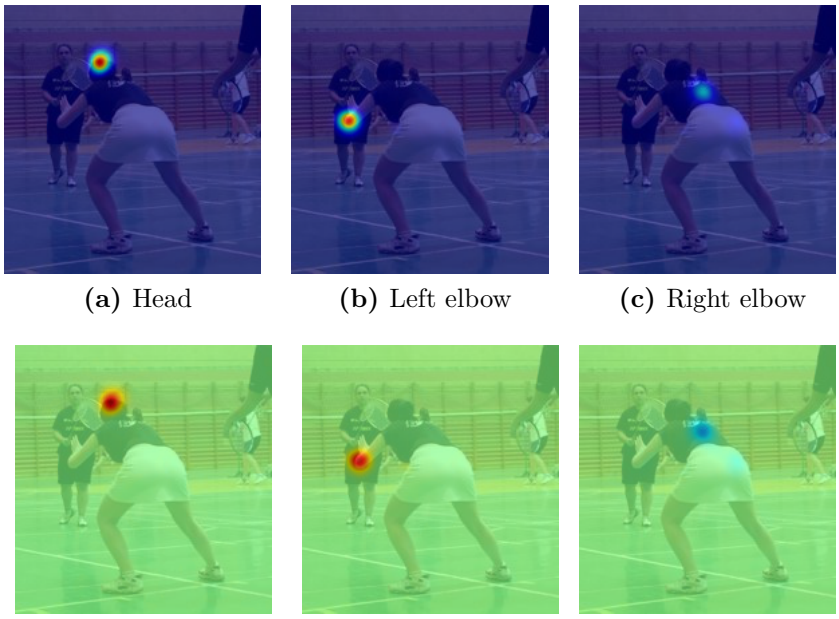
1. Determine the joint location $\hat{y}_j = \arg\max_{z \in Z} \hat{h}_j^S(z)$
2. Extract the corresponding visibility score at this location: $\hat{l}_j = \hat{v}_j(\hat{y}_j)$
3. If $\hat{l}_j$ is smaller than a fixed threshold c', label joint j as occluded, otherwise as visible.

3.2.2 Additional Network Branch

Since the CPM network needs to learn a second task, it is necessary to add a second branch that produces the required output. To limit the additional overhead, as much computation as possible should be shared with the existing network layers. Thus, the additional classification branch is intended to reuse existing network features (in the same way as stages $s > 3$ in the CPM architecture are reusing the image features from stage 2, see Figure 2.3). Since the classification task also depends on the location of joints (due to the heatmap encoding), it seems reasonable to perform classification on high-level network features that already contain information about possible joint locations. Layer `Mconv4` has to encode such features, since the subsequent output layer only performs a linear combination (1×1 convolution). The proposed classification branch consists of a single layer that is identical to `Mconv5` and is trained to produce the visibility heatmaps $\hat{\mathbf{v}} = (\hat{v}_1, \ldots, \hat{v}_J)$ for the $J = 14$ joints. No background map is generated here. This branch is only intended as an additional network output without feeding it to subsequent stages. Thus it is only added to the last stage $s = 3$. The relevant part of the extended network can be seen in Figure 3.4. The output of the additional branch $\hat{\mathbf{v}}$ is compared to the visibility ground truth $\mathbf{v} = (v_1, \ldots, v_J)$ using the same euclidean loss as for joint localization:

$$f_{vis} := \sum_{j=1}^{J} \sum_{z \in Z} \|\hat{v}_j(z) - v_j(z)\|_2^2 . \tag{3.2}$$

We refer to this modified CPM as the *VB* (visibility branch) variant in the following figures.



(a) Head **(b)** Left elbow **(c)** Right elbow

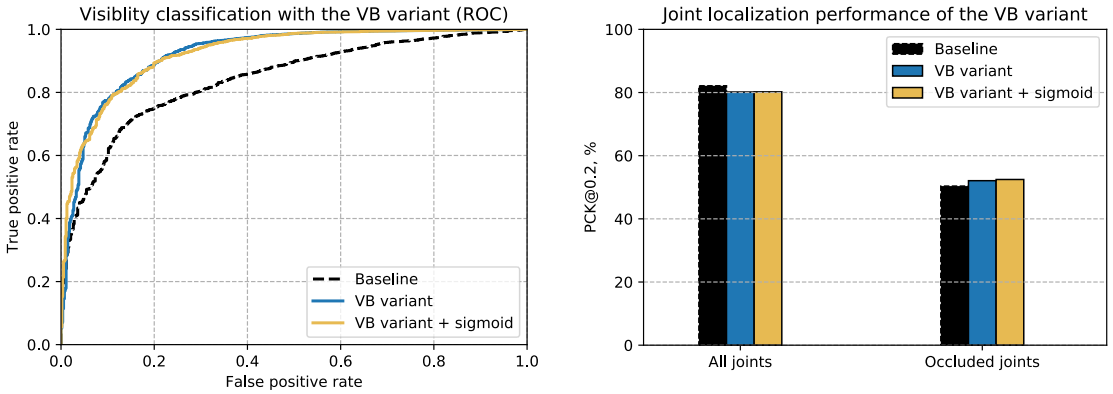
(d) Head (visibility) **(e)** Left elbow (visibility) **(f)** Right elbow (visibility)

Figure 3.5: Network output using the additional visibility branch. First row: Joint localization heatmaps for head, left elbow and right elbow. Second row: Visibility heatmaps for the same joints. Occlusion of the right elbow leads to a negative visibility score in (f) (compared to positive peaks in (d), (e)).

Only the original LSP data is used for training to avoid the imbalance of annotated and non-annotated occluded joints described in Section 3.1. All network weights are fine-tuned, starting with the default 3-stage CPM model from the baseline approach. Weights in the new visibility layer are initialized randomly. In a preliminary training for 10k iterations, all layer weights are locked except for the new layer. This is done to overcome the random initialization and decrease the visibility loss f_{vis} to an order of magnitude equal to the remaining loss terms f_s. Afterwards, all layers are trained jointly for additional 55k iterations until the loss converges.

3.2.3 Evaluation

Before we quantitatively evaluate the VB variant, the qualitative example from Figure 3.1 is revisited. Figure 3.5 depicts the output heatmaps (both for joint localization and visibility classification). Both head and left elbow are detected with a strong peak in the corresponding localization heatmaps. The visibility heatmaps contain a positive peak at the same location. The network locates the occluded right elbow close to the shoulder, which is at least close to the correct location. A negative peak at the same location can be found in the visibility heatmap for this joint. Using a visibility score threshold of $c' = 0$ (sign-based decision), all



(a) Visibility classification (ROC). **(b)** Joint localization (PCK).

Figure 3.6: Performance of the VB variant on the LSP test set. For comparison, the result of the baseline approach is shown again (stroked).

three joints would be labeled correctly. For this example, the network is able to distinguish visible from occluded joints and encode the decision appropriately in its output.

The quantitative classification result on the LSP test set is shown in Figure 3.6a. It is obvious that the VB variant dominates the baseline approach across all FPR values. For a fixed FPR value of 0.20 , a TPR of 0.90 is achieved. This is a TPR increase of 0.15 compared to the baseline approach. The improvement comes at a negligible performance overhead, because only a single 1×1 convolution layer with 14 filters is added to the network. Interestingly, this result is not obtained by a sign-based score threshold of $c' = 0$ as it was motivated. In order to reach a FPR of 0.20, a much higher threshold of $c' = 0.38$ has to be used. This hints at a notable fraction of occluded joints still receiving positive visibility scores. Figure 3.7 depicts the median and quartiles of the visibility scores across all LSP test set joints. For the baseline approach, occluded joints tend to receive lower scores, but the margin between the upper 75% of visible joints and the lower 75% of occluded joints is rather small. The VB variant is trained to assign scores of 1 and -1 to visible and occluded joints respectively. But the median for occluded joints is close to 0. Thus the network is not able to reliably reproduce the visibility encoding of $[-1, 1]$ that is anticipated during training. Still, the margin between the 75% of visible and occluded joints is widened, which leads to the aforementioned increase in classification performance.

One cause for this effect could be the euclidean loss used during training. The loss entails that the visibility heatmaps generated by the network should match the ground truth as close as possible. That is, a score of -1 in the ground truth maps has to be reached exactly to gain minimal loss. Since the visibility heatmaps are used to solve binary classification problem, the only property of interest is the sign of the scores, not their magnitude. But a score of -2 is still punished by the loss

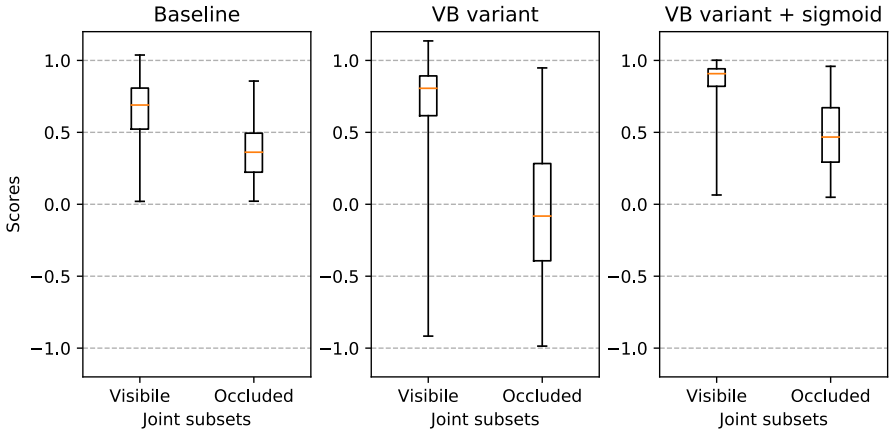


Figure 3.7: Visibility score statistics on the LSP test set joints with median and quartiles. For the baseline approach, the score from the localization heatmap is used. For the other experiments, scores are taken from the visibility heatmaps.

definition, even though the sign is correct. A common way to handle this situation is to add a sigmoid activation function, mapping higher negative scores to values close to 0 and higher positive scores to values close to 1. That way, the magnitude of the network output is not as important as the sign. Since the network output is now in the range of $[0, 1]$, the ground truth heatmaps have to be adjusted accordingly (0 encodes "occluded", 1 encodes "visible"). The loss in Equation 3.2 is changed to the negative log-likelihood (cross entropy), which is the usual choice for sigmoid activation functions to avoid saturation [GBC16, p. 183]. Figure 3.6a depicts the classification results, labeled as VB variant + sigmoid. Unfortunately, no increase of performance except in the very low FPR region can be observed. Looking at the score statistics in Figure 3.7, a large fraction of occluded joints is still not assigned the correct sign (leading to sigmoid values above 0.5). Despite the theoretical motivation for the sigmoid variant, other factors like the unbalanced number of visible and occluded joints in the training set seem to dominate the problem.

What still needs to be discussed is the effect of the network additions on the original task of joint localization. Recall that the VB variant is trained end-to-end, limiting the training examples to the original LSP data set that actually contain annotations for occluded joints. Therefore, the variation and complexity of poses is significantly reduced compared to the training of the default CPM model used in the baseline approach. Figure 3.6b depicts the joint localization results on the LSP test set. As expected, the VB variant suffers a decrease of 1.9% in PCK@0.2 performance when measured across all joints. Notably, the performance on occluded joints did actually increase by a similar amount. There are two possible causes for this effect: Either the larger fraction of occluded joints in the training set or a positive influence of explicitly detecting occlusion on joint localization.

3.3 Handling Missing Annotations

The presented visibility extension to CPMs acts as a trade-off between joint classification and localization. Visibility prediction improves significantly compared to the baseline approach, while joint localization performance decreases. In the remainder of this chapter we discuss how the VB variant can be adjusted to avoid the loss in pose estimation performance.

The VB variant is trained without the data from the Lees Sports Pose extension LSPe. It is a reasonable assumption that the vast reduction of training set variety has a negative effect on the overall performance. The reason for excluding LSPe are the missing annotations for occluded joints (as discussed in Section 3.1). Every time the network encounters a non-annotated joint during training, the corresponding ground truth heatmaps are set to zero. No matter what location or visibility the network predicts, the subsequent loss will punish the network for producing any output at all. Therefore, a solution is needed to incorporate LSPe into training while avoiding the punishment of possibly correct predictions.

3.3.1 Defining a Partial Euclidean Loss

A solution is presented in [TS14], where missing annotations do not contribute to the training loss. Although their loss definition operates on estimated cartesian coordinates, the same idea can be adapted for the heatmap-based loss in the CPM framework. We define a partial euclidean loss f'_s after stage s as

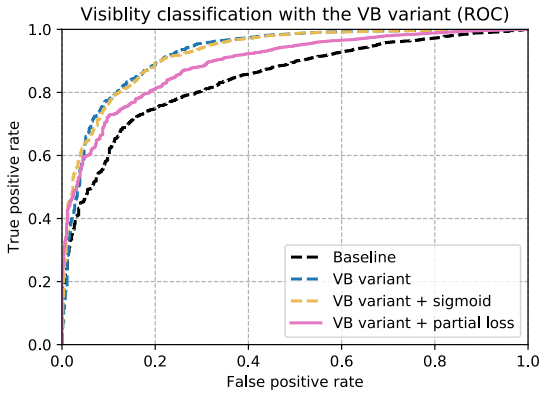
$$f'_s := \sum_{j=1}^{J+1} \sum_{z \in Z} \mathbb{1}(y_j) \cdot \left\| \hat{h}^s_j(z) - h_j(z) \right\|_2^2,$$

where $\mathbb{1}(\cdot)$ is an indicator function defined as

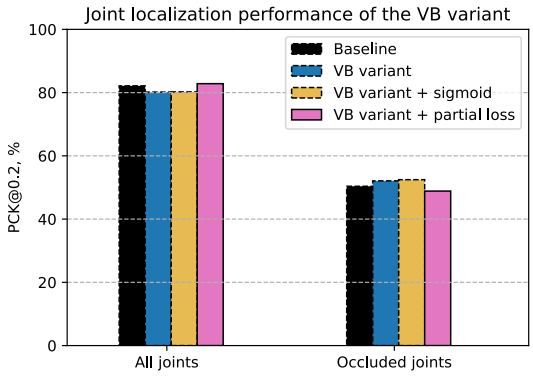
$$\mathbb{1}(y_j) := \begin{cases} 1, & \text{if } y_j \text{ is given (annotated) or } j = J+1 \text{ (background map)} \\ 0, & \text{otherwise.} \end{cases}$$

Only the output heatmaps for annotated joints and background influence the loss. This loss definition can be used for visibility heatmaps in the same way.

Consequently, we trained a third VB variant on LSP + LSPe using the partial euclidean loss. The training scheme is identical to the previous experiments except for an additional 100k training iterations due to the increased training set size.



(a) Visibility classification (ROC).



(b) Joint localization (PCK).

Figure 3.8: LSP test set performance of the VB variant with partial euclidean loss, trained on original LSP + LSPe. For comparison, the results of the previous experiments are shown again (stroked).

3.3.2 Evaluation

We show the result on visibility classification in Figure 3.8a. The ROC curve still dominates the baseline result, but with a lower margin compared to the previous experiments. At a fixed FPR of 0.20 a TPR of 0.82 is achieved, an increase of 0.07 to the baseline result. While not as good as the initial VB variants, it is an improvement none the less. Pose estimation performance is shown in Figure 3.8b. The addition of LSPe to the training indeed prevented the reduction in joint localization performance. Due to the partial loss, even a small increase compared to the baseline result can be observed.

3.4 Conclusion

In this chapter we showed how to explicitly predict the visibility of joints. By extending the default CPM network with an additional branch, this classification task can be learned alongside the original human pose estimation task. It is possible to maintain pose estimation performance while achieving superior classification results compared to the baseline CPM. The computational overhead with a single additional convolution layer is negligible. Excluding LSPe from the training process has proven to be disadvantageous for joint localization. This motivated the definition of an alternative loss to explicitly handle missing annotations in LSPe. The resulting small increase in pose estimation performance might even motivate its usage beyond the scope of this chapter.

4 Application in Sports: Pose Estimation for Swimmers

In recent years, an increasing interest in computer vision applications in the sports domain can be observed. One important reason are broadcasts of sport events being among the most popular content on TV and the internet [TSDB15]. The footage offers plenty of possibilities to gather additional statistics. This includes real-time information to enhance the viewer experience (e.g. ball possession statistics in soccer) as well as individual and team performance statistics for the participating athletes and their coaches. Yet, evaluating the footage by hand is time consuming and impractical. This motivates the need for systems that can infer the required information from image and video data in a semi or fully automatic fashion. Prominent tasks include sports type [GM13] and activity recognition [TSDB15], tracking athletes and other objects of interest in videos [SKS15, WSL+15], detecting movement over time for performance evaluation [HS16] and human pose estimation [FGH13]. [MTH15] offers an overview of a wide range of application.

The role of human pose estimation in sports is two-fold. First, multiple existing benchmark datasets for, human pose estimation (e.g. LSP, MPII) contain images from the sports domain. The higher variety and complexity of poses encountered in sports increases the difficulty of these benchmarks. Second, pose information can in turn be used for further domain-specific analysis, e.g. movement assessment to facilitate coaching and training. In this context, the following chapter describes the application of the CPM framework to human pose estimation in sports, using the example of video recordings of top-class swimmers.

4.1 CPM for Swimming Channel Footage

In competitive swimming, swimming channels can be used for individual performance analysis and improvement. These consist of a pool with an adjustable artificial water current, flowing in a fixed direction. Cameras are positioned at various locations around the pool, both above and below the water surface, to record the athlete in the channel. By matching the flow velocity, the athlete can perform normal swimming motion while staying in the same position relative to the cameras. This enables the recording of swimming motion over a long period of time. The recordings can then be used by an expert of the field to work out

possible improvements for the individual athletes. Inferring information about the athletes movement, the stroke rate or other kinematic parameters over time requires to annotate the video material. The annotations can range from a sparse selection of frames showing characteristic poses (*key-poses*) to frame-wise locations for joints or body parts [ZL15]. Especially in the latter case, annotating by hand is a tedious and time consuming task and thus limits the benefit of the recordings. A pose estimation system that automatically generates reliable pose annotations could alleviate this drawback.

4.1.1 Video Data

The video material used in this work is provided by the *Institut für Angewandte Trainingswissenschaft* (IAT) Leipzig[1]. It has been recorded with a stationary camera behind a glass pane at the left side of a swimming channel. The athletes are filmed in a side-view, partially above the water surface. The swimming direction in the recordings is always right to left. The data consist of 24 videos with 200 - 400 frames and a resolution of 720×576 pixels, recorded at 50i. Each frame shows exactly one swimmer, with all body parts located within the image bounds. The videos comprise different athletes, male and female, different flow velocities and the four different swimming styles: backstroke, breaststroke, butterfly and freestyle. Figure 4.1 shows exemplary video frames for all four styles.

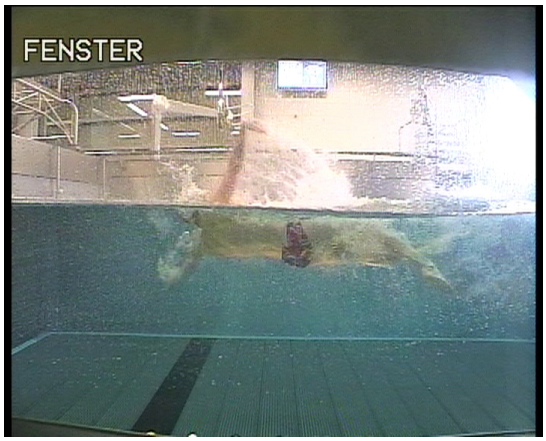
All frames are annotated using the same 14-joint person-centric annotations as in the Leeds Sport Pose dataset. For the swimming styles with symmetrical motion, i.e. breaststroke and butterfly, only the left side of the body is annotated by hand. Due to the side-view, the right side body parts are usually directly occluded by their left counterpart. Using the same 2D location for both left and right joints is thus a good approximation in most cases. For the backstroke and freestyle videos (anti-symmetrical motion), all 14 joints are annotated explicitly.

The viewpoint of the camera and the underwater setting present multiple challenges for human pose estimation:

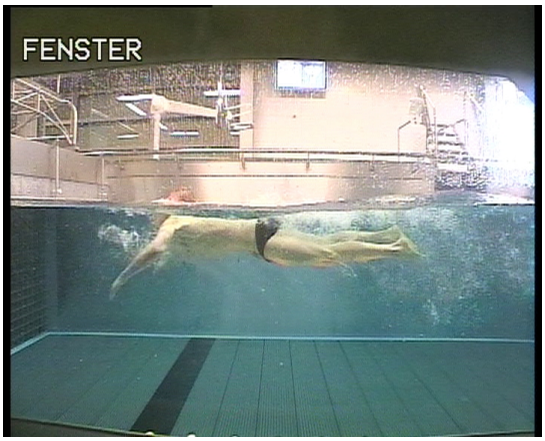
- Image noise due to bubbles and spray (Figure 4.1a).
- Refraction at the water surface can lead to ambiguous joint locations (Figure 4.1b).
- Joints can be occluded by the water surface (Figure 4.1c).
- Frequent self-occlusion (Figure 4.1d).

This makes precise joint localization difficult, even for a human observer. At the same time, the variety of poses, appearance and background is rather limited compared to single-image datasets like LSP.

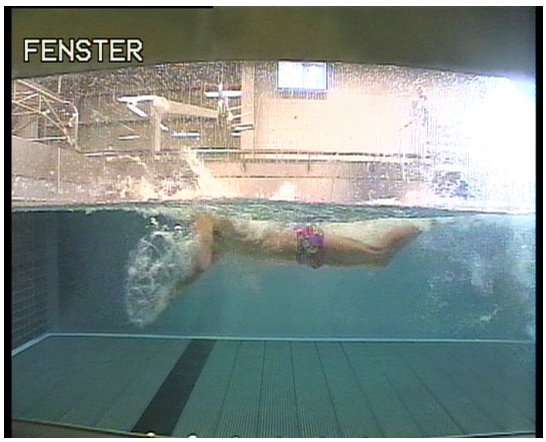
[1] http://www.sport-iat.de



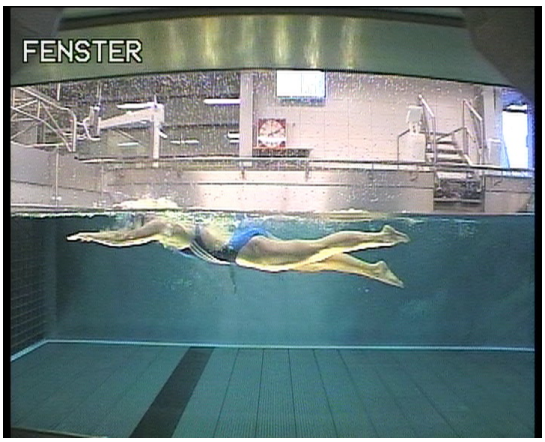
(a) Backstroke



(b) Breaststroke



(c) Butterfly



(d) Freestyle

Figure 4.1: Exemplary frames from the swimming channel footage. The data poses multiple challenges: (a) Image noise due to bubbles and spray. (b) Ambiguous joint locations (head) due to refractions at the water surface. (c) Occlusion by the water line (head, lower leg). (d) Self-occlusion (left arm).

4.1.2 Training a Baseline CPM

We train a 3-stage CPM on the swimmer data for a first impression on how well CPMs perform in this specific scenario. The data is partitioned into training and test data. The split is performed on video boundaries, such that frames from one video are either all in the training set or in the test set. Due to the cyclic nature of swimming motion, similar poses occur multiple times in a video, combined with similar visual appearance. Distributing such frames into training and test set would violate the principle of "unseen" examples in the test set [GBC16, p. 104]. Thus, one video per swimming style is held out as the test set, the remaining videos are used for training. Training and test set sizes are depicted in Table 4.1. Note that the CPM is not specialized to a specific swimming style since the training set

	Total	Backstroke	Breaststroke	Butterfly	Freestyle
Training set	6029	1765	1464	1600	1200
Test set	1117	400	317	200	200

Table 4.1: Number of swimmer images (frames) in the training and test set.

contains video frames for all four swimming styles.

The CPM framework relies on the scale and center of the person in each image. Since this information is not explicitly annotated for the swimmer videos, the same joint bounding-box is used to estimate these properties as it is done for LSP images in Section 2.3.2. The 3-stage CPM network pre-trained on LSP is used as an initial model instead of training a new CPM from scratch. That way, training time is kept feasible while possibly inheriting the benefits of the larger variety in LSP. Learning parameters are kept the same, with the exception of learning rate reduction after 50k iterations and reduced randomized image rotation in the range $[-25°, 25°]$. Rotational invariance is not as important here, since all poses share the same horizontal orientation. The network is trained for 60k iterations until no more significant reduction in training loss can be observed. We will refer to this CPM model as the baseline swimmer CPM.

4.1.3 Evaluation

The test set performance of the baseline swimmer CPM is depicted in Figure 4.2a. A detection rate of 90.1% using the PCK@0.2 metric across the complete swimmer test set is achieved. Given the challenges the swimmer footage entails, this is a notable result compared to the 82.1% on LSP. Table 4.1 reveals that the four swimming styles are not represented equally in the test set. It contains twice as many examples for back- and breaststroke compared to freestyle. Hence, the PCK score on the complete test set is biased towards backstroke and breaststroke performance. It is therefore important to consider performance on each style separately. Figure 4.2a depicts the PCK scores when limiting the test set to a specific swimming style. The results on breaststroke and butterfly are excellent with 96.6% and 97.7%, respectively. For backstroke and freestyle, only 86.7% and 79.0% of joints are detected correctly. This indicates that symmetrical and anti-symmetrical swimming styles pose challenges of varying difficulty. Evidently, pose estimation for breaststroke and butterfly examples seems to be an easier task, since left and right joints share the same annotations. Both styles implicitly pose the limited task of having to localize 8 joints only, whereas all 14 joints have to be detected explicitly for the anti-symmetrical swimming styles.

To identify strengths and weaknesses of the CPM on swimmer data, we show some test set examples and the estimated poses in Figure 4.3. Each row corresponds to one swimming style.

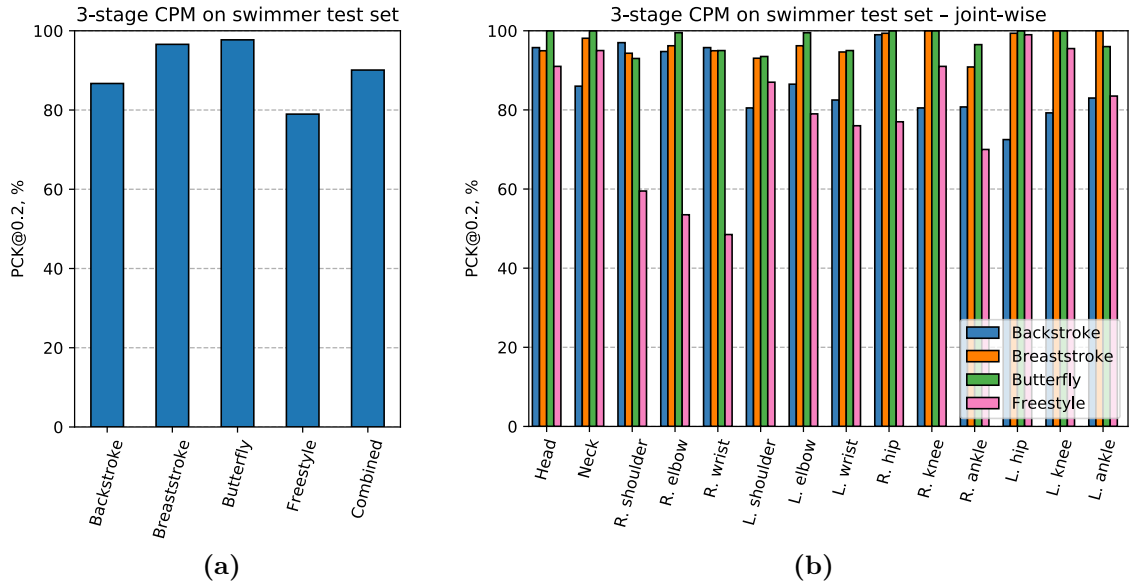


Figure 4.2: Results of the baseline swimmer CPM on the swimmer test set. (a) PCK, combined across all joints. (b) PCK for individual joints.

Backstroke

For backstroke, the CPM is capable to detect most of the joints, including occluded joints whose location has to be estimated without direct visual evidence. There are cases in which left and right parts of the body are mapped onto the same positions (row 1, column 3), mainly in presence of self-occlusion. Confusion between the left and right joints appears to be the most frequent error (row 1, column 4). Note that the limbs closer to the camera correspond to the right side of the body. This is unique to backstroke examples due to the face-up position in the water. The left-right differentiation is nevertheless correct for the majority of test set frames, the CPM seems to be able to detect backstroke examples (or equivalently detect the vertical orientation of the swimmer) and handle them appropriately.

Breaststroke

The breaststroke examples contain rarely any major failures. Erroneous localizations are usually limited to the wrists (row 2, columns 3,4). The fact that annotations for the right side of the body are identical to the left side is only an approximation. There are still frames where e.g. the left and right ankle are not located at the same image coordinates (row 2, column 4). Sometimes the CPM treats left and right joints of a breaststroke swimmer differently, which leads to a correct joint prediction w.r.t to the image content, but it does not coincide with the ground-truth annotation (row 2, column 3). However, for the vast majority of breaststroke examples the CPM has learned to predict the same locations for left and right joints, despite differing locations in the actual image. It requires the

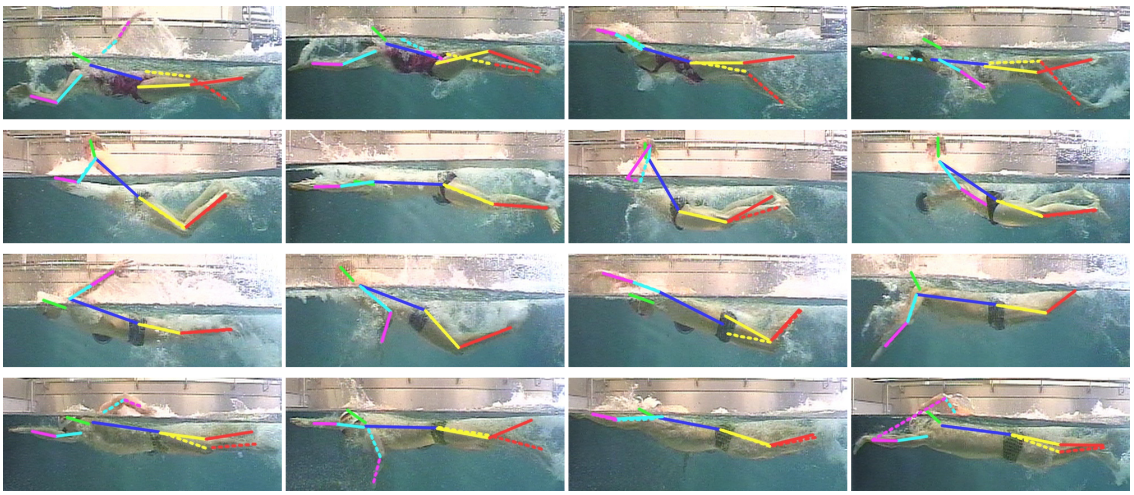


Figure 4.3: Qualitative results of the baseline CPM on the swimmer data. Each row contains examples from one swimming style: backstroke, breaststroke, butterfly, freestyle. Columns 1 and 2 show correctly estimated poses, columns 3 and 4 show partial failure cases.

CPM to treat backstroke examples differently compared to the anti-symmetrical swimming styles.

Butterfly

The same observations regarding erroneous left-right differentiation can be made for the butterfly examples (row 3, column 3). Additionally, limbs crossing the water surface can sometimes lead to truncated poses (row 3, column 4).

Freestyle

Examples of freestyle images show frequent errors regarding the arms and legs. Occlusion is not handled appropriately in many cases, leading to left and right joints being mapped onto the same positions (row 4, column 3,4).

In order to quantitatively justify the observations above, Figure 4.2b depicts the joint-wise PCK for each swimming style. Reliable localization of head and neck is given independent of the swimming style. Performance on breaststroke and butterfly is good across all joints ($> 90\%$). Hardly any difference between left and right joints can be observed here. Obviously, the CPM has learned to treat those two styles appropriately (no left-right differentiation). For backstroke, most errors occur on joints of the left arm and leg. These joints are frequently occluded. This also affects the performance on the right ankle and wrist due to left-right confusion. Similar observations can be made for freestyle, where occlusion of the right parts of the body is frequent. Performance on the right arm is especially low, with a right wrist detection rate of 48.5%.

4.2 Separate CPM Models for Different Swimming Styles

It became obvious how performance varies for different swimming style. Recognizing and differentiating between styles seems to be an essential capability the CPM needs. The examples in Figure 4.3 show that the network has indeed learned to handle instances appropriately according to the swimming style. But there are still failure cases, e.g. the freestyle example in row 4, column 3 where the right arm is occluded by the torso. The estimated pose resembles a typical breaststroke configuration. The information that the example belongs to the freestyle category would be essential to correctly estimate the location of the right arm. This motivates the straightforward idea to learn a separate CPM model for each swimming style.

The expected advantages are twofold: First, the networks no longer needs to implicitly distinguish the swimming styles. Second, all network capacity can be used to learn a single model for a specific style instead of sharing it to handle all styles appropriately at the same time. However, there is also a possible drawback of this approach: The Training set for each CPM model is limited to the 4 - 6 training videos of the corresponding swimming style. This increases the risk of overfitting by further reducing the already limited diversity in poses and appearance.

The training protocol for each swimming style model is just like the training of the baseline swimmer CPM in the previous section. Only the training iterations are limited to 20k due to the smaller training set. The loss reaches a value slightly below the loss of the baseline swimmer CPM. Further training is omitted to avoid eventual overfitting.

Figure 4.4 depicts the results of the four separate CPM models, each applied to the corresponding portion of the test set. The combined score shows the PCK across the complete test set: the pose for each test set examples is predicted using the respective swimming style model before evaluating all examples jointly. The combined result surpasses the baseline swimmer CPM by 3.3%. But again, it is necessary to look at the individual swimming styles. For the symmetric styles the results hardly change (+0.1% and −0.4% for breaststroke and butterfly, respectively). This indicates that the baseline CPM is already able to reliably identify and handle these instances. Backstroke performance is at 88.8%, an increase by 2.1%, while the freestyle result increases by notable +14.6% to 93.6%. While still below the scores for breaststroke and butterfly, the improvement for freestyle in particular shows that specializing a CPM model to one particular style can be a huge advantage. It outweighs possible negative effects of reduced training set diversity and overfitting.

Before finishing this chapter, a short study of a productive human pose estimation system in the context of swimmers is presented.

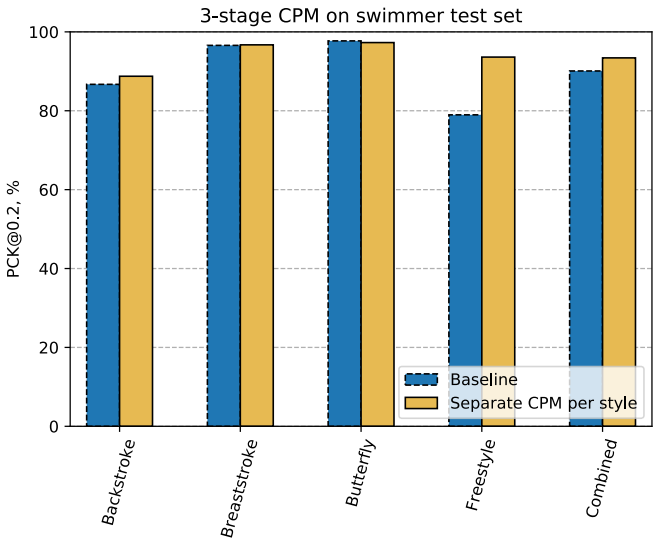


Figure 4.4: Results of four CPM models trained on separate swimming styles. The baseline result is shown again for comparison.

4.3 Considerations for End-To-End Human Pose Estimation

The CPM framework is intended to estimate the pose of a person in a 2D image. It requires the knowledge where the person is roughly located. More specifically, it needs estimates for the center and the scale of the person relative to the image size. This information is used to crop and scale the image such that it has a specific quadratic size and is centered around the person. When imagining a productive system for automated pose annotation in the swimming channel scenario, this additional information is not available. The size of athletes in the recordings varies, depending on the shape and size of the body and the distance between the swimmer and the camera. The horizontal location of the athletes can also deviate from the center of the camera. Before applying the CPM, a rough detection of the athlete is therefore required for each video frame.

Detecting and classifying objects is a long-standing research topic in computer vision. Similar to human pose estimation, convolutional neural networks have been used extensively for this task during the last years. Popular approaches are for example RCNN [GDDM14] and its successors [Gir15, RHGS15] and SSD [LAE+16]. These systems are trained to detect objects of different categories along with estimated bounding boxes.

The idea is now to combine such an object detector with the CPM framework to obtain an end-to-end system for human pose estimation applicable to the swimming channel footage. In this case, the Single Shot Detector (SSD) is used. It is trained on the simmer data to detect athletes and provide a tight bounding box es-

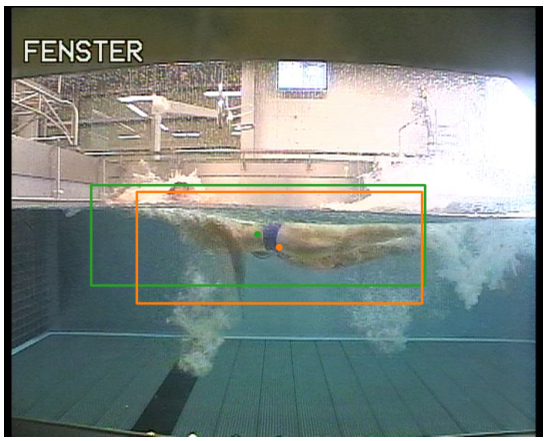


Figure 4.5: Comparison of bounding boxes used to estimate scale and center of the person: Tight box around joint annotations used in CPM training (green); Bounding box proposed by SSD (orange). Colored dots indicate the center of each bounding box.

timate. The derived scale and center of the athlete can then in turn be used by the CPM for pose estimation. We conduct an experiment with the freestyle CPM from Section 4.2. Pose estimation is performed on the freestyle instances of the test set using the bounding box estimates computed with SSD. This leads to a PCK@0.2 of 93.1%, a small decrease compared to the 93.6% when using the freestyle CPM with the tight bounding boxes around ground truth joint locations. Thus, using estimated bounding boxes during evaluation can lead to a loss in performance. When comparing the ground truth bounding boxes and the ones computed with SSD, both are rather similar for the majority of test set instances. But there are exceptions, where SSD fails to capture the whole person (e.g. by truncating an arm, see Figure 4.5). In these cases, the CPM is used on bounding boxes that were not present during training, a possible explanation for the decrease in performance.

To avoid this mismatch between CPM training using ground truth bounding boxes and evaluation on SSD proposals, the training data should be enriched appropriately. For each video frame in the training set, a second instance is added, paired with the bounding box estimate using SSD. Additionally, random flipping of images during training is disabled, because the swimming direction is constant in the present use case. The CPM network can now learn to cope with different bounding boxes, while specializing on poses with a predominant orientation. All remaining learning parameters are identical to the freestyle CPM training in the preceding section.

When evaluating this variant on the freestyle test set using SSD bounding boxes, a score of 94.5% is achieved, an increase of 0.9% compared to the original result of the freestyle CPM. This shows that training enrichment using varying bounding boxes and specialization regarding the orientation can further improve performance for SSD-based pose estimation.

4.4 Conclusion

It was shown that the CPM framework is applicable to the specialized task of human pose estimation for swimmers. Compared to general human pose estimation datasets like Leeds Sports Pose, the side-view footage from a swimming channel poses quite different challenges with overall less variation but more image noise and frequent occlusion. Apparently, the CPM is able to handle noisy image content due to the excellent results on breaststroke and butterfly instances. The notably lower performance for the anti-symmetric swimming styles showed that occlusion and the necessary differentiation between left and right joints lead to frequent errors. Training separate models for each swimming style alleviates the CPM network of learning a general model applicable to all styles. This lead to a notable performance improvement, given that the swimming style is known for each instance. The preceding section gave a quick review of how two off-the-shelf systems can be combined to construct a standalone system for human pose estimation in the context of swimming channel recordings.

5 Class Labels as Multimodal Input and Output

In the previous chapter it was shown that different instances of human pose estimation problems – in this case athletes in different swimming styles – pose varying challenges to the CPM framework. We argued that the network had to implicitly learn to distinguish instances of different styles. Distinguishing between swimming styles explicitly by learning separate models limited to one swimming style led to a notable increase in performance.

The additional information about the swimming style partitions the instances into four distinct classes. Each video and thus each frame is labeled to belong to one of these classes. In this chapter, the question that will be explored is how swimming style information can be handled explicitly inside the CPM network. More generally, how can the existing CPM architecture be extended to incorporate multimodal information: categorial class labels and spatial-continuous pose predictions.

In Section 5.1, swimming style information will be used as additional input for the CPM. The goal is to learn a single CPM model that competes in performance with the single-style models from the previous chapter.

Section 5.2 covers the inverse situation: using the CPM framework to predict the swimming style. Additionally, the sequential nature of video frames will be utilized to further improve performance for this classification task.

Section 5.3 combines the ideas of the preceding sections. It presents an architecture with favorable pose estimation performance that does not require swimming style labeling but infers this information on its own.

5.1 Utilizing Swimming Style Information

Knowing what swimming style an athlete performs can be seen as a form of contextual information that is available alongside the swimming channel footage. The idea to use the context of a 2D image or its depicted scene to improve human pose estimation has already been investigated, partially in the sports domain. In [SKN10], the information of athletes interacting with other objects is used, with the example of basketball or soccer players interacting with a ball. By explicitly modeling and detecting the ball, detections for the interacting part of the body

(arm or leg) and thus the whole pose are improved. In the scope of swimmers in a swimming channels, the context is not some additional object, but rather the information that some poses are more likely than others (e.g. when comparing typical poses for butterfly and breaststroke in Figure 4.3).

The goal is to alter the CPM architecture to use the swimming style information for each image as additional input. That way it might be possible to incorporate the separate single-style CPMs into one combined model while maintaining their pose estimation performance. In fact, there is even the hope to surpass their performance since the combined model can be trained on the complete training data with more variety in appearance, background and noise. The analysis can also provide insight into the cause why the single-style models perform notably better compared to the baseline swimmer CPM: the additional style information or the multiplied network capacity.

The practical advantage of such a combined CPM is that only a single network model has to be stored. Additionally, when new training data for a particular style becomes available, the performance on all styles might benefit from it.

5.1.1 Encoding Class Labels in Convolutional Neural Networks

Before additional (multimodal) information in the form of class labels can be added to the CPM network, it is necessary to model and encode it appropriately. There have been multiple studies on how to incorporate multimodal data in convolutional neural networks. For example, in [TDB16] a 2D image and spherical coordinates of a different camera viewpoint are used to synthesize a new view of the scene. They use an encoder-decoder network architecture to find an abstract representation for the image content and simply concatenate it with a learned representation of the input coordinates. The CPM network however is a fully convolutional network and thus requires a compatible encoding for the additional input.

5.1.1.1 One-hot Class Label Maps

We propose an encoding of class labels using additional input channels, one for each class. Each of these class label maps is of size 46×46, the internal size the CPM network operates on after repeated pooling. The class label maps are one-hot encoded, e.g. the first map is filled with ones while the remaining three maps are zeroed to encode a “backstroke”-label. These additional maps can now be added as additional input to one or multiple convolution layers in the network (similar to the center-map discussed in Section 2.3.3.3).

Before specifying a concrete architecture utilizing the additional input, it is necessary to take at closer look at the chosen encoding. More specifically, the question whether a convolution layer can actually utilize such additional input has to be answered.

Imagine a convolution layer with a single filter and multiple input channels, followed by a ReLU activation function. Class label maps are now added as additional input to the layer. Thus, the layer can now learn appropriate filter weights that operate on the class input. For a fixed class label, the label maps are constant (all are zeroed except for one). After training the layer, convolving the constant maps with the fixed filter weights leads to a constant term added to the final filter output. In fact, this term is equal to the sum of weights operating on the one-filled map. The weights on the remaining label maps are multiplied by zero and thus do not contribute to the output. Hence, the only effect the class maps have is to shift the filter output by a constant value. Taking into account the subsequent activation function, this constant value is basically shifting the zero-threshold of the ReLU. Depending on the label, the ReLU output is therefore either increased, decreased or zeroed when falling below the threshold. One possible effect could be that the convolution layer learns a switching functionality: Only if a specific label is given, the filter activation surpasses the ReLU threshold and produces a non-zero output.

This observation merely acts as a motivation, it remains to be seen if the convolution layers will indeed learn such a behavior.

5.1.1.2 Adding Class Label Maps to the CPM Network

The benefit of swimming style information is that it reduces the variety of poses to expect. For a backstroke example, it is less likely that both wrists are positioned close to each other. Instead, a positioning in opposite directions relative to the shoulders is more probable. Therefore, the style information can be used to learn more specific dependencies between joints. Adding the swimming style input to stage 1 in the CPM network might thus not be useful, as it operates on local image content with a limited receptive field. The subsequent stages seem more appropriate, as they have the capability to learn long-range spatial dependencies between different joints and can naturally benefit from the additional input.

It is unclear though, which layer will benefit the most or if it is necessary to repeatedly include the class label maps into the network. To this end, two different experiments are conducted. In the first setting the CPM network is altered such that class label maps are added once at the beginning of each stage $s \geq 2$, right before layer `Mconv1`. Figure 5.1 depicts the relevant modifications. This version is denoted “Style input - single concat”. In the second experiment, the class maps are additionally included as input for all subsequent layers. This will be referred to as “Style input - repeated concat”.

In both cases the network is trained using the complete set of the swimmer training data, alongside the swimming style label for each examples. Network weights are initialized with the 3-stage CPM model trained on LSP from Section 2.3.2, additional weights due to the new input maps are initialized randomly using a gaussian

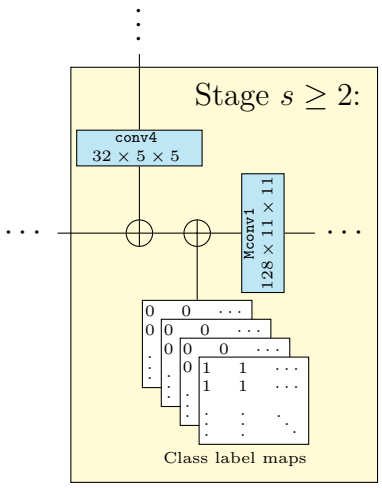


Figure 5.1: "Single concat" CPM architecture using class label maps as additional input. Only the relevant layers are shown.

distribution with zero mean. The remaining learning parameters are identical to the baseline swimmer CPM in Section 4.1.2.

5.1.2 Evaluation

We evaluate the two variants of swimming style input CPMs on the swimmer test set and discuss the effect of the newly added class label maps on network activations.

5.1.2.1 Human Pose Estimation Performance

Figure 5.2 depicts the pose estimation performance of both versions of the swimming style input CPM on the swimmer test set. The single concat variant performs slightly better than the single-style CPMs on all swimming styles except freestyle, for which a loss of -6.8% PCK can be observed. Still, the model dominates the baseline result across all styles. The repeated concat variant performs similar: The improvement on backstroke, breaststroke and butterfly is slightly smaller, but so is the performance loss on freestyle with only -2.0%. Combined over the whole test set, both variants are on par with the single-style models. Detailed results are listed in Table 5.1.

The results show that it is indeed possible to combine the single-style models into one general CPM without a major loss in PCK. The additional swimming style information seems to be the dominating cause for the increased performance of the single-style models. However, the anticipated benefit of training the style input CPM on the complete training set is only partially confirmed. The improvements compared to the single-style CPMs, if any, are rather small.

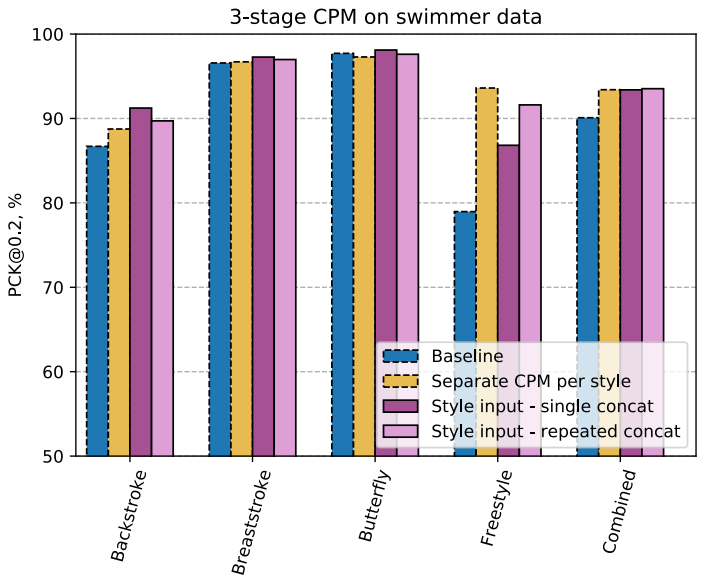


Figure 5.2: Results of the swimming style input CPM variants on the swimmer test set. For convenience, results of the baseline and single-style CPM models are shown again (stroked).

	Backstroke	Breaststroke	Butterfly	Freestyle	Combined
Baseline CPM	86.7	96.6	97.7	79.0	90.1
Single-style CPMs	88.8	96.7	97.3	**93.6**	93.4
Single concat	**91.2**	**97.3**	**98.1**	86.8	93.4
Repeated concat	89.7	97.0	97.6	91.6	**93.5**

Table 5.1: PCK@0.2 of the swimming style input CPM versions "Single concat" and "Repeated concat" on the swimmer test set. The results of the single-style CPMs and the baseline swimmer CPM (without swimming style information) are shown for comparison.

The convolutional neural network is apparently able to utilize the additional class label using the one-hot label map encoding. For this reason it is worth to take a closer look at how the additional input affects the network layers.

5.1.2.2 Effect of Class Label Maps on Filter Activations

As illustrated before, for a given swimming style the four class label maps act as a constant input for the subsequent convolution layer. The filters in this layer contain weights applied to the additional label maps. If the weights for one input map are in total > 0, the filter output is increased whenever this label map is "activated" (the corresponding label is given). Otherwise if the sum is < 0, the output is decreased or even set to zero due to the ReLU activation function. In all other

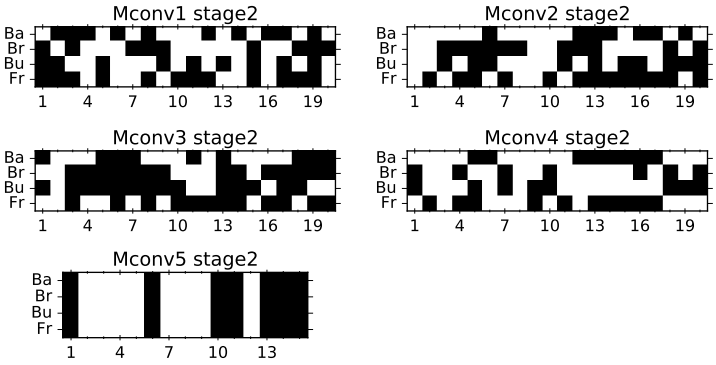


Figure 5.3: Sign of summed filter weights applied to the label maps for the first 20 filters in the layers of stage 2 of the repeated concat CPM variant. Signs are color coded: white for positive, black for negative. For each layer, columns correspond to filters 1 - 20, rows correspond to label maps – backstroke, breaststroke, butterfly, and freestyle, from top to bottom.

cases, i.e. the label map being inactive (zeroed) or a weight sum of 0, the filter output is not influenced in any way by this particular label map. The sign of the summed weights on each label map thus decides, wether the filter reacts positively, negatively or simply ignores the label information.

We want to observe if there are learned filters that depend on the label. Figure 5.3 depicts the sign of the summed weights on the individual label maps for stage 2 layers in the repeated concat variant of the CPM. Only the weights for the first 20 filters in each layer are shown to avoid a cluttered visualization. The sign is color coded: white for a positive sum of weights, black for a negative sum of weights. Each column shows the weight signs for one filter on the four possible labels backstroke, breaststroke, butterfly and freestyle (top to bottom). Single-color columns hint at a filter that reacts similar to any given swimming style. Mixed-color columns represent filters that mainly activate on some, but not all styles. The majority of filters in the first four layers evidently distinguish between different label values. Notably, the last layer seems to be indifferent with respect to the label. This is to be expected, because the filters in the last layer always have to produce an equally scaled output (a gaussian heatmap) for each joint, no matter what swimming style is given. Note that this sign-based representation of filter weights only shows qualitatively that in many cases the filter output does depend on the label, but not to what extend.

We conduct another experiment to analyze how much filter responses depend on the class label . The same input image is forwarded twice through the repeated-concat CPM, once with the correct backstroke label, the second time with a deliberately incorrect breaststroke label. For both cases, the activations of the first four filters in layer `Mconv1_stage2` are depicted in Figure 5.4. Filter 1 shows a lot of activations in presence of the backstroke label, but only very few when a breaststroke label

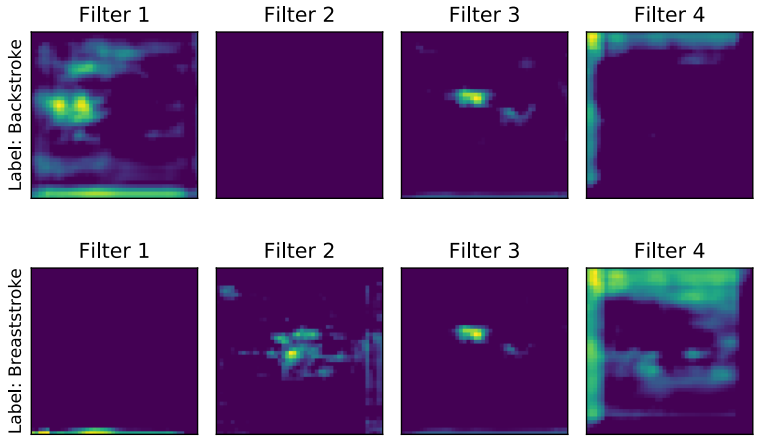


Figure 5.4: Filter activations in layer `Mconv1_stage2`. First row: Activations using a swimmer image and backstroke label as network input. Second row: Activations using the same image but a breaststroke label.

is given. This coincides directly with the weight signs for this filter in Figure 5.3 (top left visualization, first column): Only the backstroke label map is weighted positive. The opposite case can be seen for filter 2, which seems to mainly activate on breaststroke examples. Filter 4 behaves similar, but additionally has positive weights on the butterfly and freestyle label maps. Filter 3 shows close to equal activations in both cases. Its weight signs in Figure 5.3 indicate that it mainly reacts to butterfly examples, all other label maps are weighted negatively. But as noted before, negative weights on a label map do not necessarily result in no activations when this label is present. Concluding, it can be noted that the filters do indeed learn a switching behavior as anticipated. This can be on/off switching like filter 2 in the example, or just a reduction of activation for some labels as in filter 4.

Based on the weight sign analysis above, Figure 5.5 depicts for each swimming style how many filters have positive weights on the corresponding label map and thus are more likely to "activate" on these instances. Especially layers 2 to 4 in both stages have more filters activating on the anti-symmetric swimming styles. The network has learned to specialize more of its filters to those styles. This coincides with our impression that human pose estimation on backstroke and freestyle examples is more difficult and requires the network to use more of its computational capacity (in the sense of convolution filters) on these swimming styles.

5.2 Swimming Style Estimation

We showed how contextual information can improve human pose estimation. Swimming style information reduces the overall complexity of the problem by providing

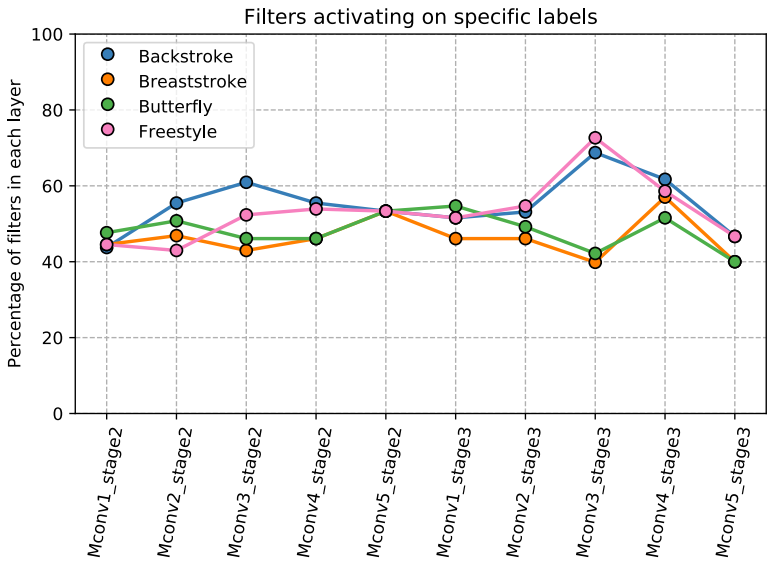


Figure 5.5: Percentage of filters that activate on the different swimming style label values, for all layers in stages 2 and 3 of the repeated concat CPM variant.

additional clues on which poses are more likely to observed. But even when this information is not available, the pose estimation system still has to handle instances of each swimming style differently, as it was argued for the baseline swimmer CPM in Section 4.1.2.

In this section, an inverse scenario is considered: For each video of the swimming channel recordings, the swimming style of the athlete is unknown. Instead, it is required as additional output. The question is how the CPM framework can handle pose estimation and classification of a person's pose or activity simultaneously. Similar to Chapter 3, this is an instance of multitask learning where a system has to solve multiple tasks while sharing computations or learned features between these tasks. For neural networks, multitask learning can even improve performance on the individual tasks, e.g. due to regularization effects of shared features or representations suitable to solve all tasks simultaneously [Car96]. Predicting joint visibility is only an extension to the existing task of joint localization. Here, true multitask learning is required since pose and swimming style estimation are two closely related but altogether different tasks regarding their level of abstraction.

5.2.1 Extended CPM Architecture for Human Pose Estimation and Classification

While swimming style estimation is a very specific task, it can be seen as an instance of either image classification or activity recognition. The latter subsumes tasks where it is necessary to infer what activity (e.g. movements or interactions)

one or multiple persons are performing. In the domain of sports, this can be high-level activity descriptions like playing basketball or soccer [GM13] or more detailed information like what action a soccer player is currently performing [TSDB15]. Because activities often consist of movement over time, a time series of images or other sensor recordings is often used as input data. For now, swimming style estimation should be based on single images only, which thus resembles the traditional task of image classification.

There are many examples for successful image classification with convolutional neural networks, based on the seminal work in [LBBH98] and [KSH12]. The main architectural concept consists of repeated convolution and pooling, followed by multiple fully-connected layers. The last layer typically consists of one output unit per class. A subsequent softmax function can be used to obtain a valid class probability distribution [GBC16, p. 184]. The idea is now to extend the CPM network by an additional classification branch in a similar fashion to provide pose and swimming style estimates at the same time.

The swimming style can mainly be inferred by the pose of a swimmer. Therefore, the join locations estimated in the CPM network should act as the main indicator for the resembled style. This motivates reusing high-level network features containing the appropriate information, similar to Section 3.2. The proposed architecture uses an additional classification branch in the last stage, operating on the pooled features of layer `Mconv4`. These features have to be recombined to abstract from single joints to the complete pose configuration and thus the swimming style. However, it is unclear how much additional computation is necessary. Two different variants of the classification branch are evaluated: The first consists of one additional convolution layer and max pooling, followed by a fully-connected layer with one output per class and a softmax mapping. We show the relevant modifications of the network in Figure 5.6a. The convolution layer is used to recombine the given joint features (also spatially) and reduce the number of inputs for the fully-connected layer. The second variant uses more capacity (see Figure 5.6b): Two convolution layers and max pooling followed by two fully-connected layers.

During training, all instances are of the form $(x, \mathbf{y}, l)$,where x and $\mathbf{y}$ are the image and the joint annotations as before and $l \in \{1, 2, 3, 4\}$ is the swimming style label. The softmax output $\hat{p}$ of the classification branch is trained to be a probability distribution $p(l = c)$ with

$$p\left(l = c\right) \quad = \quad \begin{cases} 1, & \text{if } c \text{ is the correct label} \\ 0, & \text{otherwise.} \end{cases}$$

The predicted swimming style label $\hat{l}$ is then simply the class with the highest probability:

$$\hat{l} \quad = \quad \underset{c \in \{1,2,3,4\}}{\arg\max}\, \hat{p}(l = c).$$

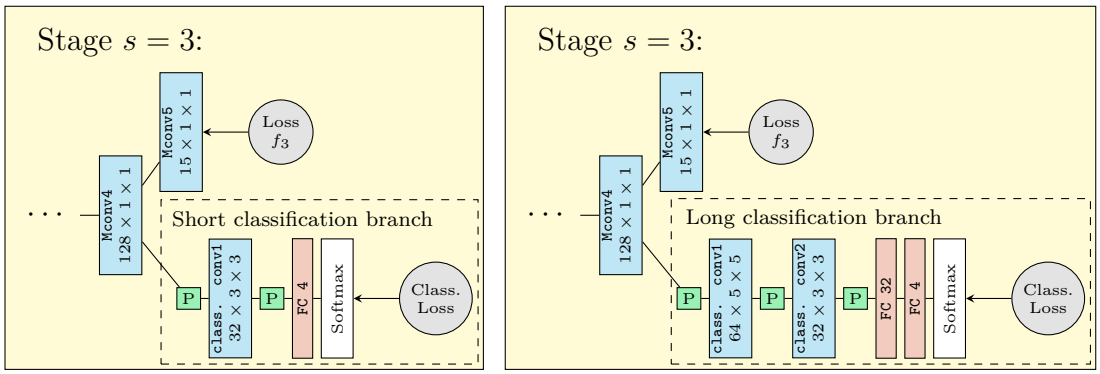


Figure 5.6: Two variants of the CPM classification branch for swimming style estimation. Only the relevant layers are shown. (a) Short classification branch. (b) Long branch with additional convolution, pooling and fully-connected layers.

The swimming style prediction CPM is trained on the swimmer training set. Training for the classification branch is performed using a multinomial logistic loss, which is scaled by a factor of 10 to compete with the remaining euclidean loss terms. The network weights are initialized using the baseline swimmer CPM from Section 4.1.2. Weights for the layers in the classification branch are initialized randomly according to a gaussian distribution with zero mean and are trained exclusively in a preliminary training for 8k iterations. An increased learning rate compared to the baseline setting is necessary to enable sufficient learning progress. Following the initial training, all network layers are trained jointly for additional 3k iterations, using the baseline learning parameters, but the additional decrease in classification loss is marginal.

5.2.2 Leveraging Sequential Swimming Style Estimates

Before evaluating and discussing the two variants of the swimming style estimation CPM, the idea to include sequential information, i.e. data over time, is revisited. Until now, frames from the swimming channel recordings have been processed independently one at a time. Even if the pose of a swimmer can be estimated perfectly, the swimming style can not always be inferred from this information alone. For example, the pose in Figure 4.3, row 2, column 2, could belong to a breaststroke or butterfly swimmer. To resolve such ambiguities, it might be necessary to look at motion over time. This train of thoughts will be followed in detail in Chapter 6. For now, a much simpler approach will suffice.

The swimming channel videos that are used throughout this work show one athlete using the same swimming style over the full length of the video. For each frame in a video it is known that all past and future frames are labeled the same. This

knowledge can now be used to improve the style estimate for the current frame, e.g. by using the estimates from preceding frames.

When the pose and style for each frames in one video are estimated, every frame at time $t = 1, \ldots, T$ is processed independently by the CPM. $T \in \mathbb{N}$ is the length of the video in frames. For each frame, the network produces an estimated probability distribution $\hat{p}_t(l = c)$ over the swimming style classes. It is now possible to average the estimates from the last k frames:

$$p_t^* (l = c) \quad = \quad \frac{1}{\max(k, t)} \sum_{\substack{i = t - k + 1 \\ i \geq 1}}^{t} \hat{p}_i (l = c) ,$$

for a fixed $k \in [1, T]$. The intention is to increase classification performance by overcoming single wrong predictions or predictions with low confidence.

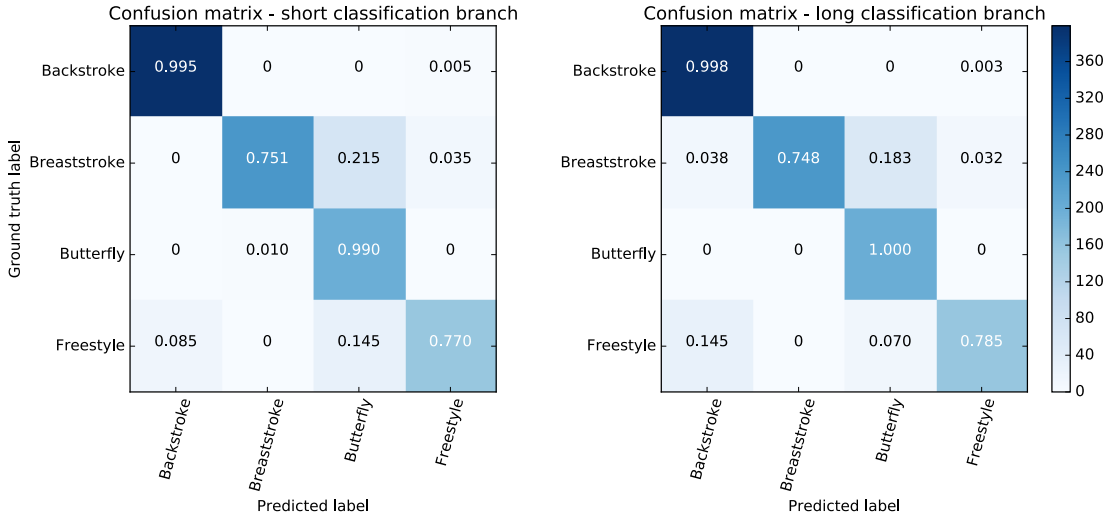
5.2.3 Evaluation

As a metric for this multi-class classification task, per-class *Precision* is used. The precision for binary classification is defined as

$$\text{Precision} \quad := \quad \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \in [0, 1]$$

and measures the fraction of instances labeled positive that are actually positive [DG06]. It can be adapted for multi-class classification by measuring the precision for each class separately. To compare classification results on all classes simultaneously with a single score, we additionally use the (non-weighted) *Average Precision* (AP) across all four classes, which is simply the mean over all per-class precision values.

The CPM for swimming style estimation is evaluated on the swimmer test set. Figure 5.7 contains the results of the short and long classification branch, without using sequential estimates ($k = 1$). It shows for each class the fraction of instances assigned a specific label in a *Confusion Matrix* . The entries on the diagonal are the per-class precision values. Both versions achieve close to perfect results on the Backstroke and Butterfly instances. Breaststroke instances are occasionally mistaken for butterfly (symmetrical styles). Freestyle is mixed up with either butterfly or backstroke. The differences between the two variants are small, with an AP of 0.885 and 0.894 for the short and long versions, respectively. At the same time, pose estimation performance stays the same as for the baseline CPM, with differences $< 0.5\%$. Thus, the CPM is indeed capable of handling the swimming style classification task using shared computations and features while maintaining performance for the original task.



(a) **(b)**

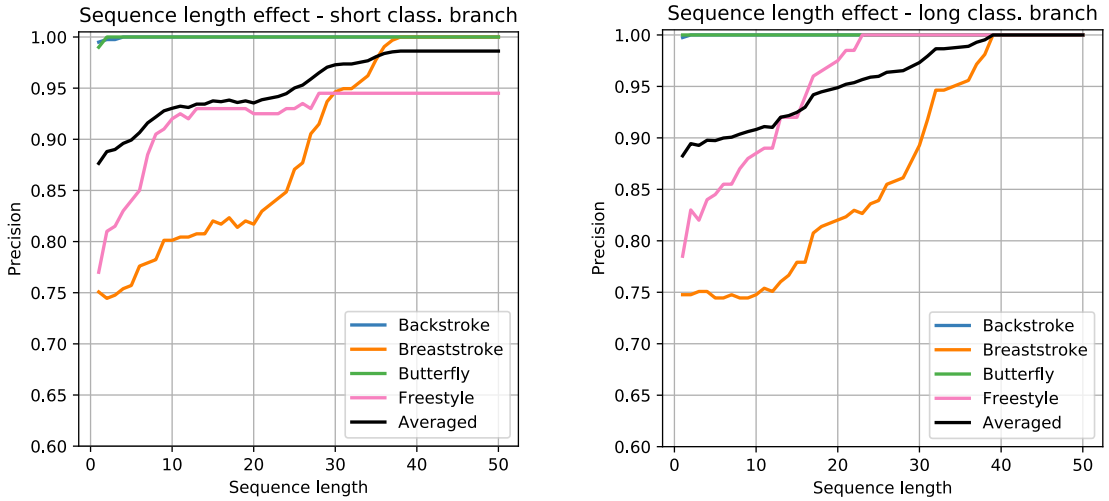
Figure 5.7: Confusion matrix showing the classification result of the swimming style estimation CPM. (a) CPM with short classification branch. (b) CPM with long classification branch.

When the sequential nature of video frames is leveraged as described in the previous section, classification performance can be increased notably. Figure 5.8 depicts the effect of the sequence length k on the classification results. For both variants, the precision on all classes increases nearly monotonically with increasing k. The simple averaging mechanism over past estimates seems to work as anticipated. The results converge for sequence lengths between 35 and 40. Using $k = 40$, an AP of 0.987 and 1.0 for the two versions is obtained. The long classification branch variant thus achieves perfect classification on the test set, at the cost of additional computational overhead compared to the short version.

5.3 Simultaneous Swimming Style Input and Estimation

The previous sections showed how pose estimation for swimmers with the CPM framework can be improved when the swimming style is known. On the other hand, the swimming style can be reliably inferred when it is not available, under the assumption of sequential video frames. It raises the question, whether both approaches can be combined: Estimate the swimming style and in turn use it to improve pose estimation. This removes the necessity that the swimming style is known for each swimming channel video.

Of course, annotating video clips with a single label regarding the swimming style



(a) **(b)**

Figure 5.8: Effect of sequence length when averaging swimming style estimates over sequential frames. (a) CPM with short classification branch. (b) CPM with long classification branch.

is by far less overhead than manually annotating joints in each frame. Still, making this task superfluous enables more automation and is a further step towards a standalone pose estimation system for swimmers.

5.3.1 Combination of Class Input and Output Architectures

The presented architecture is again intended for sequential frames, where the swimming style does not change throughout a video. The idea is to construct a CPM that during evaluation uses an estimated style as input and produces a possibly refined estimate. Since all video frames are processed in chronological order, the estimated style from the previous frame constitutes the input (besides the current frame itself). The network then provides an estimate for pose and swimming style for the current frame.

The proposed network architecture is simply the combination of the single-concat variant of the class input CPM from Figure 5.1 and the additional (short) classification branch from Figure 5.6a. For reliable estimates, the input swimming style is obtained using averaging across past style estimates with a fixed sequence length $k = 40$ as described in Section 5.2.2.

This CPM variant faces two additional challenges. Firstly, at the beginning of each video there is no estimated style when processing the first frame. Thus, the style input for the network is empty. Secondly, the estimate from previous frames can be wrong. With both cases, a simple identity mapping from input to output style

is not sufficient. The network has to rather use the input style as a hint, but still needs to be able to correct it if necessary. Therefore, the network has to be trained to handle these cases appropriately.

Because of this reason, the swimmer training set is altered. For every style, 5% of the training instances are picked randomly and are assigned an either empty or incorrect label. This should hopefully compensate for missing and wrong swimming style labels during evaluation. Training is split into three phases. First, the class-input and pose estimation parts of the network are trained equal to the scheme in Section 5.1.1.2. Second, the classification branch is trained separately, followed by a short joint training of all network layers. This is identical to the training of the short classification branch in Section 5.2.1. The training sums up to a total of 71k iterations, starting with the layer weights from the CPM model trained on LSP.

5.3.2 Evaluation

The CPM is again evaluated on the swimmer test set. Averaging over the last $k = 40$ sequential style estimates leads to a close to perfect classification result, with an AP of 99.2%. The network is apparently able to overcome the empty style label at the beginning of each test set video. Figure 5.9 depicts the pose estimation performance labeled as "Style inferred", compared to all previous results on the swimmer data. The results are nearly identical to the single-concat CPM that uses swimming style annotations. Small losses for backstroke, breaststroke and butterfly can be observed. Combined across all styles a PCK of 93.0% is achieved, only 0.4% below the single-concat result.

Reducing the sequence length used for averaging style estimates to $k = 5$ leads in a slightly lower AP of 98.1%, but it has no negative influence on the pose estimation performance. This makes the system applicable to continuous video streams or videos where the athletes switch between swimming styles, since the transition phase until the system detects a change of style should be rather small.

5.4 Conclusion

This chapter showed how context information can be used in CPMs with the example of swimming styles. Different architectural extensions to the CPM were proposed to include class labels as additional input or infer additional information by adding a second classification task to the learning objective. The results indicate that a single CPM model using explicit swimming style information can compete with and even outperform four individual CPMs trained exclusively on one style. Without swimming style annotations, it is possible to reliably infer this information using a multitask CPM. This leads to a combined architecture that does not need style annotations while maintaining similar pose estimation performance. One key

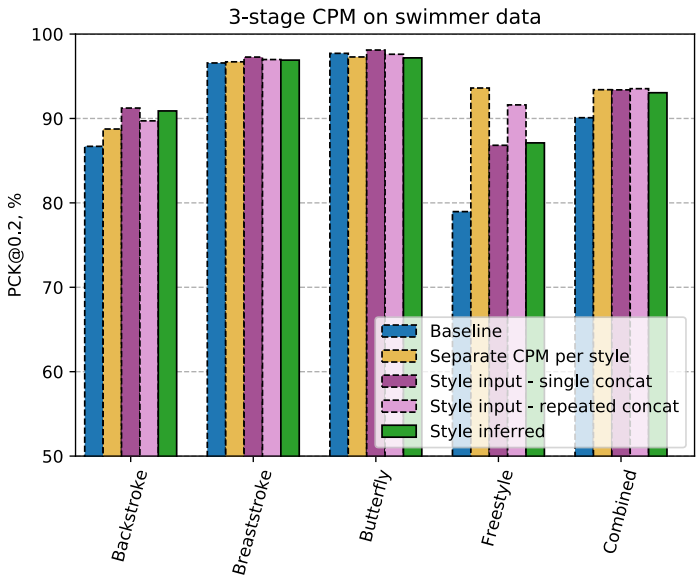


Figure 5.9: Results of the combined swimming style input and estimation CPM on the swimmer test set, labeled as "Style inferred". For convenience, results of the previous CPM variants are shown again (stroked).

aspect was to utilize sequential video frames. This idea will be the main focus of the following chapter.

6 Continuous Pose Estimation on Videos

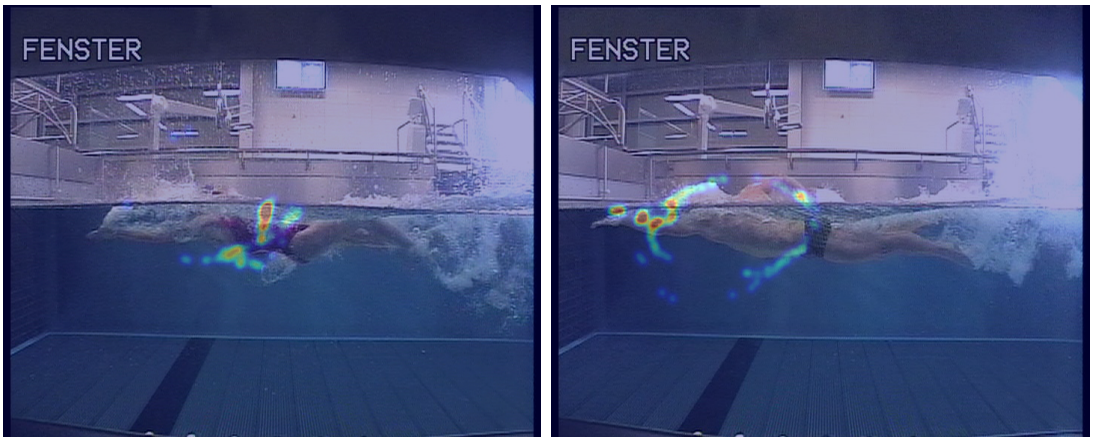
Throughout the previous chapters, pose estimation on the swimmer data is performed in a single-frame fashion: Only one video frame is processed at a time by the CPM, leading to estimates that are solely based on the visual evidence of the current frame. The exception to this is the sequential swimming style estimation in Section 5.2.2. Since the swimming style is the same for all frames of a video, it is possible to average over estimates of subsequent frames. This leads to a notable improvement.

A similar observation can be made regarding the pose. While the pose of an athlete is not constant, the changes from one frame to another are limited. This is especially true for the swimmer videos recorded at 50Hz. Thus, there is a strong dependency between the pose in the current frame and poses in past and future frames. If e.g. past poses are known, they can in turn act as an important clue for what pose to expect in the next frame(s). This benefit also became obvious when the swimmer videos were annotated in the first place: Whenever joints were not directly visible, it was necessary to step forward and backward in time to estimate the movement of joints over time. That way, locations of joints could be interpolated for the frame in question.

Given the challenges the swimmer videos pose, precise estimation of joint locations on single frames can be difficult. Sequential video frames enable humans to estimate the location of e.g occluded joints with high accuracy. Therefore, sequential information promises to be a huge benefit for human pose estimation. This chapter is thus dedicated to continuous pose estimation on videos, using the specialized example of swimming channel recordings. We discuss how information from subsequent video frames can be incorporated into the CPM framework.

6.1 Error Analysis on Swimmer Videos

Before the idea of continuous pose estimation is further developed, we take a closer look at expected benefits and challenges. The ground truth for the swimmer data was generated for each video frame by first annotating joints that could be uniquely identified and located, before the missing annotations were added by interpolating between past and future frames. Our goal is to apply a similar concept to CPMs:



(a) (b)

Figure 6.1: Heatmaps showing the ground truth locations of joints that are erroneously localized by the baseline swimmer CPM from Section 4.1.2. (a) Left wrist errors on the backstroke test set video. (b) Right wrist errors on the freestyle test set video. For reference, the heatmaps are superimposed on the first frame of each video.

First, we use the standard CPM architecture to predict the pose for each frame separately. Second, the pose estimates from subsequent frames are processed jointly to improve the estimates and eliminate errors in the single-frame predictions. It is therefore important to analyze how these errors evolve over time.

We use the baseline swimmer CPM as a reference for single-frame pose estimation. Occlusion – both self-imposed and by the water surface – was identified as a major challenge, especially for the anti-symmetrical swimming styles. But occlusion is usually limited to specific periods of the cyclic swimming motion, e.g when an arm is located behind the torso. This could imply that errors on related joints are also limited in time. For this purpose, Figure 6.1 visualizes where errors occur on the backstroke and freestyle test set videos. This is done for the left and right wrist respectively, both frequently occluded and not reliably localized (see Figure 4.2b). Figure 6.1a illustrates the locations of errors for the left wrist in the backstroke evaluation video. It appears that errors on this joint are limited to the part of cyclic motion where the left arm is partially or completely hidden behind the torso. The error locations for the right wrist on the freestyle video in Figure 6.1b reveal a different situation. This joint poses similar challenges compared to the left wrist for backstroke. But obviously, errors are not limited to one specific period of motion. It implies that errors on this joint could persists for a long duration, making continuous pose estimation based on single-frame pose estimates more difficult.

In order to get an impression how long errors persist, Figure 6.2 depicts the duration of errors on the individual swimming styles. Each time a wrong joint estimate occurs in the video, the number of frames is counted until the joint is localized

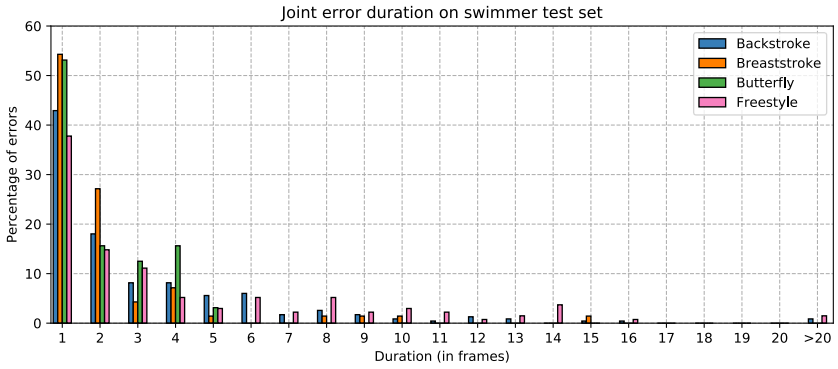


Figure 6.2: Percentage of joint localization errors persisting over a given number of frames. Locations are estimated using the baseline swimmer CPM. Results are shown for each test set video (one for each swimming style) separately.

correctly again. This is counted as one error with the given duration. The figure contains the percentage of errors with a duration of 1 to 20 frames. For the symmetrical swimming styles, errors on individual joints are limited to few sequential frames. Thus in many cases, it could even suffice to add the previous and next frame to a pose estimation system to correct that error. For backstroke and especially freestyle, erroneous joint estimates can persist for longer video sections. A notable portion of errors over 5 and more frames can be observed. Even errors over more than 20 frames are present. As a reference, one motion cycle takes about 60 - 80 frames for the backstroke and freestyle videos. Thus, there are joints the CPM localizes wrongly for complete sections of cyclic swimming movement.

The analysis shows that single-frame predictions can contain errors on the same joint for long periods of time. Because our goal is to develop a continuous pose estimation system that eliminates such errors, it needs to be able to integrate information from even longer sections of the video.

6.2 Sequential Pose Refinement

Applying artificial neural networks to sequential data is an active field of research. They are successfully used in multiple applications including speech recognition, translation and automatic captioning of images [GBC16, p. 410]. The basis for these approaches are *recurrent neural networks* (RNN). They operate on a sequence of inputs with one datum per discrete time step. The processing of one datum uses computed state information or features from the forward pass in the previous time step. That way, information can be passed from one time step to the next. The output of such networks can either be sequential too, or just one single output after the complete input sequence is processed. A popular specialization of RNNs uses the *long short-term memory* (LSTM) model. It has the ability to store and delete

information after varying time intervals (compared to the fixed step-wise information propagation in the basic RNNs) *[GBC16, p. 409]*. However, RNNs are a vast topic on its own and thus beyond the scope of this work. Additionally, gradient-based learning of RNNs requires a time-unfolding of the recurrent network: for each time step a separate instance of the network has to be created, multiplying the required memory for intermediate and output data. The input sequence is processed forward in time, before the calculated gradients can be propagated backwards in time. This can induce a large overhead in computation time and memory. Adding a hypothesized recurrent mechanism to a CPM network would even further increase the required resources of the multi-stage architecture and thus might not be feasible at all.

In the context of human pose estimation, [PCZ15] propose the *Flowing ConvNet* architecture for pose estimation on videos where fixed-length frame sequences are used to improve the estimates on the individual frames. It consists of a CNN similar to a 2-stage CPM that is trained to produce joint location heatmaps for single video frames. Additionally, a second off-the-shelf network is used to estimate the optical flow between subsequent video frames. This is used as an estimate of motion to warp the estimated joint locations in time. Finally, a third network with a single layer is learned to predict the final pose for each frame using the estimate for the current frame itself as well as the warped predictions from past and future frames. This is denoted *temporal pooling*. It leads to an overall architecture of three distinct networks that are trained separately, without any form of explicit recurrence between network layers. The architecture achieves a significant improvement compared to other single-frame human pose estimation approaches. The major drawback is the overhead incurred by the optical flow estimation. It increases the average computation time for each video frame by a factor of 10. Yet, the architecture shows that continuous pose estimation is possible without embedding an explicit recurrent mechanism.

6.2.1 Post-Processing Network for Sequential Pose Refinement

With this in mind, we develop a first extension to the CPM framework for continuous pose estimation. The stage-wise architecture of the CPM is intended to facilitate learning of spatial dependencies. In the same manner, one can imagine an additional CPM stage to learn temporal dependencies. Inspired by the temporal pooling mechanism in the Flowing ConvNet, we propose a stage that uses the estimated poses on past and future video frames to improve the pose for the current one. To avoid explicit recurrence, this stage is actually a separate network operating on sequences of pose estimates. It acts as an additional post-processing network that refines pose estimates over time.

Our overall architecture for human pose estimation on videos consists of a default 3-stage CPM and the sequential post-processing network that we now describe. We

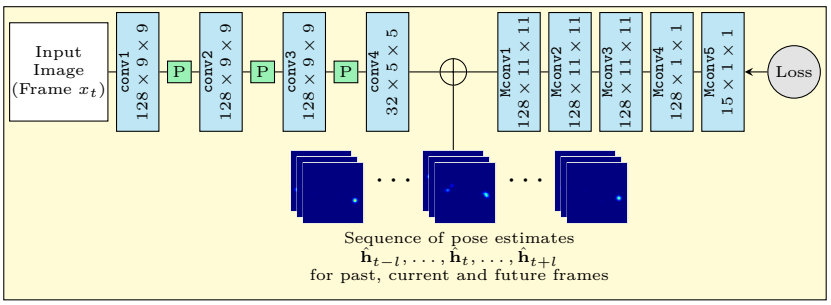


Figure 6.3: Separate post-processing network to refine sequential pose estimates. It resembles a standard CPM stage but additionally uses the single-frame pose estimates from past and future frames that are produced beforehand by a default 3-stage CPM. It is trained as a separate network to output a refined pose estimate for the current frame at time t.

denote the length of a video as $T \in \mathbb{N}$. Each video consists of frames $(x_1, \ldots, x_T)$. We apply the default CPM to each frame x_t, $t \in [1, T]$, and obtain a single-frame pose estimate in the form of localization heatmaps $\hat{\mathbf{h}}_t = (\hat{h}_{t,1}, \ldots, \hat{h}_{t,15})$ for the 14 joints and the background. Our post-processing network uses frame x_t and the *sequence of pose estimates* $\mathbf{z}_t$ with

$$\mathbf{z}_t \quad := \quad (\hat{\mathbf{h}}_{t-l}, \hat{\mathbf{h}}_{t-l+1}, \ldots, \hat{\mathbf{h}}_t, \ldots, \hat{\mathbf{h}}_{t+l-1}, \hat{\mathbf{h}}_{t+l}) \tag{6.1}$$

as input. It outputs a refined estimate $\mathbf{h}_t^*$, which is again a set of $14 + 1$ joint localization heatmaps for the frame at time t. The idea is that the estimates from past and future frames provide a strong evidence where joints have to be located in the current frame, e.g. by inferring joint movement over time. $l \in \mathbb{N}$ is a free parameter that defines how many single-frame estimates from past and future frames the network uses for pose refinement. The length k of the input sequence $\mathbf{z}_t$ is thus $k := 2l + 1$.

The architecture of the post-processing network is depicted in Figure 6.3. It is identical to a default CPM stage with the same layers and the same number of convolution filters. The only difference is the additional heatmap sequence, added right before layer `Mconv1`. Note that the input heatmaps $\hat{\mathbf{h}}$ are the result of an explicit scale search by averaging over multiple CPM forward passes with varying images scales (see Section 2.3.3). No additional scale search is intended for our post-processing network, however.

6.2.1.1 Generating Training Data

We now want to train and evaluate the presented post-processing network on the swimmer videos. Each training example from the swimmer training set has to be of the form $(x_t, \mathbf{y}_t, \mathbf{z}_t)$, with the current video frame, its ground truth joint

annotations and the estimated joint localization heatmaps for the past, current and future frames, respectively. While x_t and $\mathbf{y}_t$ are available, the heatmaps have to be generated first. During evaluation, these heatmaps are the potentially erroneous output from a single-frame CPM. This has to be the case for the training, too. The straightforward approach would be to apply the baseline swimmer CPM to the swimmer training data and store the resulting joint localization heatmaps. These can then be used to train the post-processing network. But when applying the baseline swimmer CPM to the same data it was trained on, the results are overly optimistic. In fact, for the swimmer training set a PCK of 99.9% is achieved. Thus, the CPM estimates contain close to no errors. This would lead to a major deviation between training and test data for our additional network. In order to resolve this problem, the original swimmer training data is split in half. While training the baseline swimmer CPM on one half, it is applied to the other half and vice versa. That way, estimated joint localization heatmaps for the complete swimmer training set are obtained that are more likely to resemble the estimates encountered later during evaluation.

Each training example for the post-processing network contains 14+1 heatmaps for every past, present and future frame within its input sequence length. With a sequence length of k, this add up to $15 \cdot k$ heatmaps of size 46×46. In the definition of the original CPM, training includes efficient data augmentation by altering the input image and the joint annotations. Here, data augmentation would also entail scaling, rotating and flipping all additional input heatmaps, resulting in a vast overhead. No data augmentation is thus used for training. This of course severely limits the variety in training data. It remains to be seen what effect it will incur.

The layer weights of the post-processing network are initialized with the stage-3 weights of the baseline swimmer CPM. This is possible since both use the same architecture. New weights in layer `Mconv1` (due to the increased input size) are initialized randomly according to a gaussian distribution with zero mean. In a preliminary training, only this layer is trained for 1k iterations to overcome the random initialization. Then the complete network is trained for additional 5k iterations, which suffices for the loss to converge.

6.2.1.2 Evaluation

We expect that the sequence length parameter k has a direct influence on pose estimation (refinement) performance. As a first experiment, the post-processing network is trained with $k \in \{1, 3, 7\}$. The case $k = 1$ can be seen as a verification experiment. With not sequential information at all, the network simply resembles a fourth stage in the usual CPM framework. [WRKS16] argue that additional stages up to a total of $s = 6$ are beneficial, at least when evaluated on LSP. The performance for this additional fourth stage should accordingly reach at least the baseline swimmer CPM performance. Figure 6.4a depicts the PCK results of all three versions, evaluated on the the swimmer test set. No matter which sequence

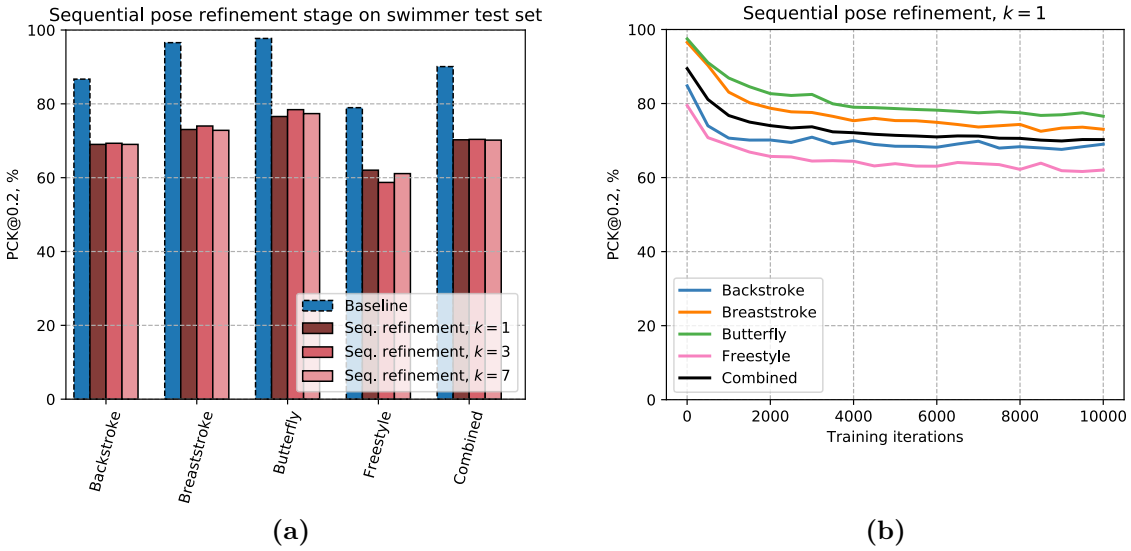
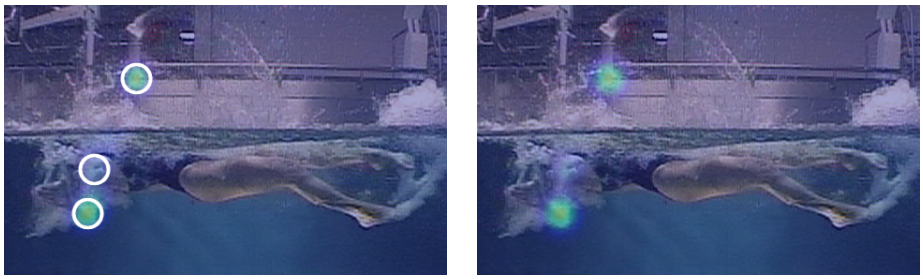


Figure 6.4: Results of the post-processing network for sequential pose refinement on the swimmer test set, trained without data augmentation. (a) Performance using different sequence lengths. Baseline swimmer CPM performance is shown for comparison. (b) Performance for sequence length $k = 1$ w.r.t. training duration.

length is used, the result is far below the baseline, with a decrease of about -19% on the complete test set. However, the additional sequential information does not seem to be the cause: even the version with $k = 1$ performs equally bad. Thus, either the network architecture or the training scheme have to contain a serious flaw. For closer inspection, Figure 6.4b depicts the test set performance of the $k = 1$ variant as training progresses. It reveals a substantial overfitting effect as soon as the training begins: While training loss decreases, far below the loss of the baseline swimmer CPM, test set performance decreases as well. Interestingly, the initial performance (after 0 iterations) where network weights are identical to the stage-3 weights of the baseline model is on par with the baseline performance. This means that in general, the additional fourth stage is not the culprit. The training scheme, and presumably the missing data augmentation, lead to the observed deterioration of pose estimation performance.

6.2.2 Efficient Data Augmentation

It is now necessary to enable data augmentation for training, but avoid the vast overhead of scaling and rotating possibly hundreds of heatmaps for each training example. In the original CPM framework, efficient data augmentation is possible since ground truth heatmaps (for loss calculation) are generated on the fly. The only overhead inflicted by data augmentation is scaling, rotating and flipping the



(a) (b)

Figure 6.5: Predicted heatmaps from a single-frame CPM can contain activations at multiple locations. Reducing the heatmaps to a set of sparse 2D points should preserve such ambiguities. (a) Output heatmap for the left elbow using the baseline swimmer CPM. Possible joint locations are identified by local maxima in the heatmap that surpass a fixed minimal threshold. A small circular region is used, shown in white. All three locations are stored with the respective heatmap score. (b) When training the post-processing stage, the heatmap is recreated with gaussian blobs at the stored locations.

training image and the annotated joint locations, i.e. 2D points.

One way to enable efficient data augmentation for the additional stage is to reduce the dense heatmaps in the input sequence $\mathbf{z}_t$ to single 2D joint locations. For this it is important to observe that the output produced by the single-frame CPM is trained to encode predicted locations with a blob resembling a gaussian distribution at the corresponding position in the heatmap. If there are ambiguities between joints the network cannot resolve, the heatmaps can contain multiple detections and thus multiple gaussian-like activations. Figure 6.5a illustrates an example of this case. To preserve such ambiguities for a subsequent post-processing, it is necessary to keep all predicted locations and not just the one with the highest score.

For this purpose, the training heatmaps for the post-processing stage are searched for local maxima in a small region to find the peaks of any gaussian blob. If the score at this location exceeds a fixed minimal threshold, the location and score are stored. The size of the local region and the threshold are chosen by hand. The heatmaps are then discarded. During training, each training example can be augmented by rotating, scaling, etc. the video frame and the extracted joint locations for all frames within the sequence length. Afterwards, the actual heatmaps are recreated by placing a gaussian blob in the heatmaps for each predicted joint location. The gaussian is normalized such that it has the stored score at its maximum. The spatial size of the gaussian controlled by the standard deviation is again chosen by hand, such that the original heatmaps and the recreated version closely resemble each other. Figure 6.5b depicts this process for the previous example.

Using this process, the sequential training data is generated anew as described in Section 6.2.1.1. As an additional precaution, the original swimmer training

data is split in not only two, but in five distinct sets: one for evaluation/heatmap extraction, the remaining for training the baseline CPM. This is done to ensure that the single-frame pose estimates the post-processing network operates on are of similar quality for training and evaluation.

6.2.3 Effect of Sequence Length

With data augmentation enabled, the sequential post-processing stage is trained, again with varying sequence lengths up to $k = 13$. Figure 6.6a depicts the results on the swimmer test set, alongside the baseline performance without sequential refinement. It is obvious that the previous drop in performance is no longer present. This shows that data augmentation is an important prerequisite for successful learning in the CPM framework, especially when working on data with low variability. The verification experiment with $k = 1$ reaches and slightly surpasses the baseline performance. It confirms that an additional and separately trained stage even without any sequential input is a viable option. When the stage does use sequential pose estimates, i.e. $k > 1$, performance on the combined test set is steadily improving. For $k = 13$, a combined PCK of 91.7% is achieved, an additional +1.6% compared to the baseline result. This improvement is almost entirely caused by backstroke and freestyle, with +3.3% and +1.2% on those styles respectively. As a first experiment, this shows that the proposed stage can improve single-frame estimates and that longer frame sequences are advantageous as anticipated in Section 6.1.

6.2.3.1 Sparse Sequences

Because performance gradually increases when more sequential information is passed into the network, the question arises if this is also true for even longer sequences. But simply adding more and more pose estimates from past and future frame might not be an option, as it increases network input tremendously. Each additional frame in the input sequence means an additional 15 input heatmaps. With a sequence of $k = 13$, the network already has to process and learn to utilize 180 additional heatmaps. However, it is not obvious if including pose estimates for all frames in the regarded input sequence is indeed necessary. With a frame rate of 50Hz, the difference between one frame in the swimmer videos and the next is limited. This includes the change in joint locations, especially since the joint localization heatmaps in the network are 8-times pooled.

For this reason, further variants of the post-processing network are evaluated using longer input sequences but reduced temporal resolution by skipping heatmaps from each second frame. More formally, with a sequence length of $k = 2l + 1$ we restrict the heatmaps in the input sequence $\mathbf{z}_t$ of the network to

$$\mathbf{z}_t \quad := \quad (\hat{\mathbf{h}}_{t'} | t' \in [t - l, t + l]\,,\; t' \equiv t \pmod 2). \tag{6.2}$$

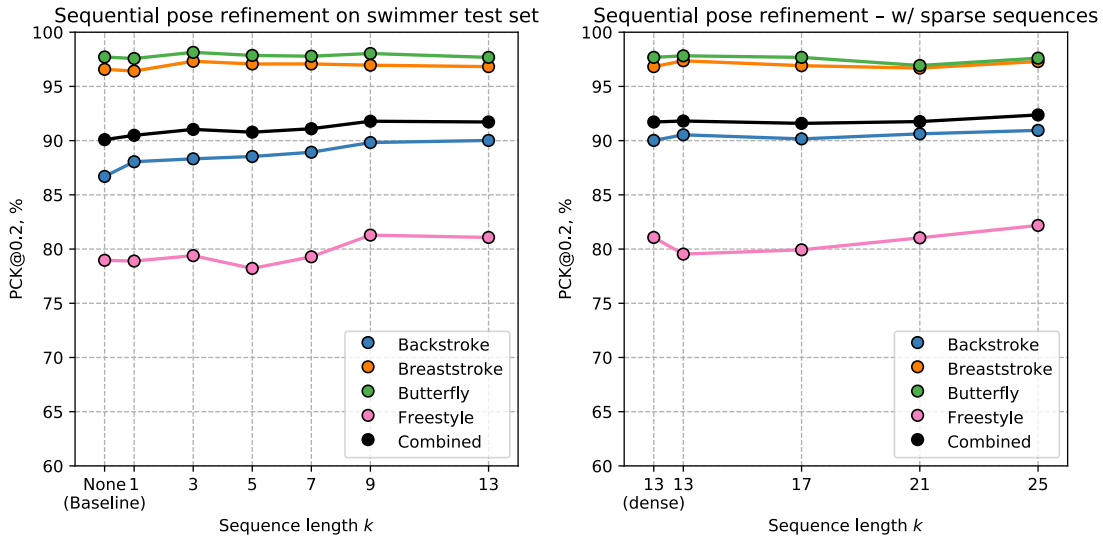


Figure 6.6: Effect of the sequence length k on the sequential post-processing network trained with data augmentation. (a) Dense input sequences of length 1 - 13. The baseline result without sequential refinement is shown for comparison. (b) Sparse input sequences of length 13 - 25, including only pose estimates from every second frame.

We show the result for $k \in \{13, 17, 21, 25\}$ in Figure 6.6b. Dividing the number of input heatmaps for $k = 13$ in half does not reduce performance except for freestyle, where errors are more frequent and thus more input might be necessary to overcome those. When the sequence length is further increased to $k = 25$, another slight improvement of +1% in PCK for backstroke and freestyle is achieved, compared to the previous $k = 13$ variant with dense sequences. The number of input heatmaps is identical for both variants, such that the improvement comes at no additional computational overhead. The experiment shows that longer but sparse sequences are advantageous for pose refinement on the swimmer data.

6.2.3.2 Utilizing Past and Future Pose Estimates

With the best setting of $k = 25$ and frame skipping, a combined PCK of 92.4% is achieved. This result is still outperformed by the CPM models in Section 5.1 that use swimming style information. The expected advantage of sequential information is only partially confirmed. While the proposed network does utilize the additional information, it is unclear to what extent. A possible drawback of the presented network architecture is its initialization. Before training starts, the weights from the last stage of the baseline swimmer CPM are copied. This last stage uses only the current frame and the pose estimate from the previous stage. Thus, it has learned

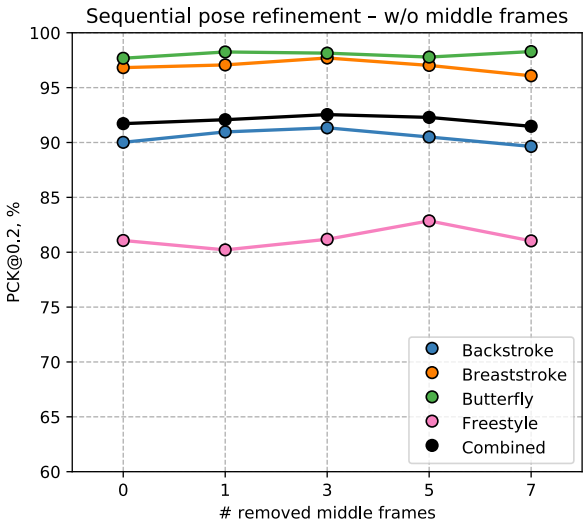


Figure 6.7: Performance of the sequential post-processing network when forced to utilize past and future pose estimates. Estimates for the current and surrounding frames in the middle of the sequence are gradually removed from the input.

to rely solely on this single estimated pose. The same behavior can be expected for the initial post-processing network. During training, the network has to unlearn this reliance and instead take advantage of the additional sequential information. For comparison, we additionally trained the network from scratch, but it revealed no significant differences in performance. This makes it less likely that the network initialization is a huge issue here.

We conduct another experiment to find out if the sequential pose information could be leveraged to a greater extent. Using a sequence length of $k = 13$, the estimated pose $\hat{\mathbf{h}}_t$ in the center of the input sequence $\mathbf{z}_t$ (for the current frame at time t) is removed. Additionally, poses for surrounding frames in the middle of the sequence are also gradually removed. The idea is to force the network to rely on pose estimates further in the past and future and observe the influence on performance. Figure 6.7 depicts the result when removing $1, 3, 5$ or 7 mid-sequence frames. Surprisingly, removing the single-frame estimates for the current and surrounding frames actually increases combined test set performance by up to $+0.8\%$ in PCK. When more input is gradually removed, performance declines again. This shows that the current architecture of the post-processing stage might not be optimal. If not forced to, the network apparently does not fully utilize the potential of the additional sequential information.

6.3 Enforcing Temporal Interpolation of Joint Predictions

The expected benefit of sequential pose estimates is that the change in pose over time can be used to estimate movement. This enables to interpolate joint locations in time and thus to predict the joints in past or future frames. Combined with the image information, this could help to resolve ambiguities and enable more stable pose estimates over time. The main challenge lies in past and future poses being estimates itself, which in turn can hinder reliable movement estimation.

In the previous approach to sequential pose refinement, all information is provided to the network at once. It has to learn what information to select and utilize by itself. It became apparent that forcing the network to use past and future frames can be advantageous. This motivates a different network architecture where we enforce prediction and interpolation of joints locations over time using only past or future information.

6.3.1 Network Architecture with Temporal Pooling

The proposed architecture in this section is based on the three findings of previous experiments: long ($k = 25$) but sparse input sequences (by skipping pose estimates from each second frame) and encouraging the network to actually utilize the provided sequential information. For the latter, the network input is divided into three parts: the single-frame pose estimates for past frames, the estimates for future frames and the estimate for the current frame together with the video frame itself. All three parts are processed in different network branches. Each branch is trained to predict the same target – the pose for the current frame. The network architecture is illustrated in Figure 6.8. The "past" and "future" branches do not possess any image information and are thus forced to predict the current joint locations based on the joint estimates in the preceding or subsequent frames. This requires to infer some notion of movement over time. The "present" branch does not have any sequential information and resembles a normal stage in the CPM framework. The outer branches have an additional convolution layer and larger filter kernels. The reason is that in long sequences, joint movement over time can cover a substantial area of the input window. Inferring movement in the complete sequence therefore requires a large receptive field. With the given architecture, these branches have a receptive field equal to half the input window, which covers most joint movement during $k = 25$ frames.

The final layer of the network is intended to integrate the predictions of all three branches. It is based on the idea of temporal pooling in the Flowing ConvNet [PCZ15]: For each joint separately, a single 1x1 convolution filter is learned on the respective heatmaps from all three branches. Each filter computes a weighted

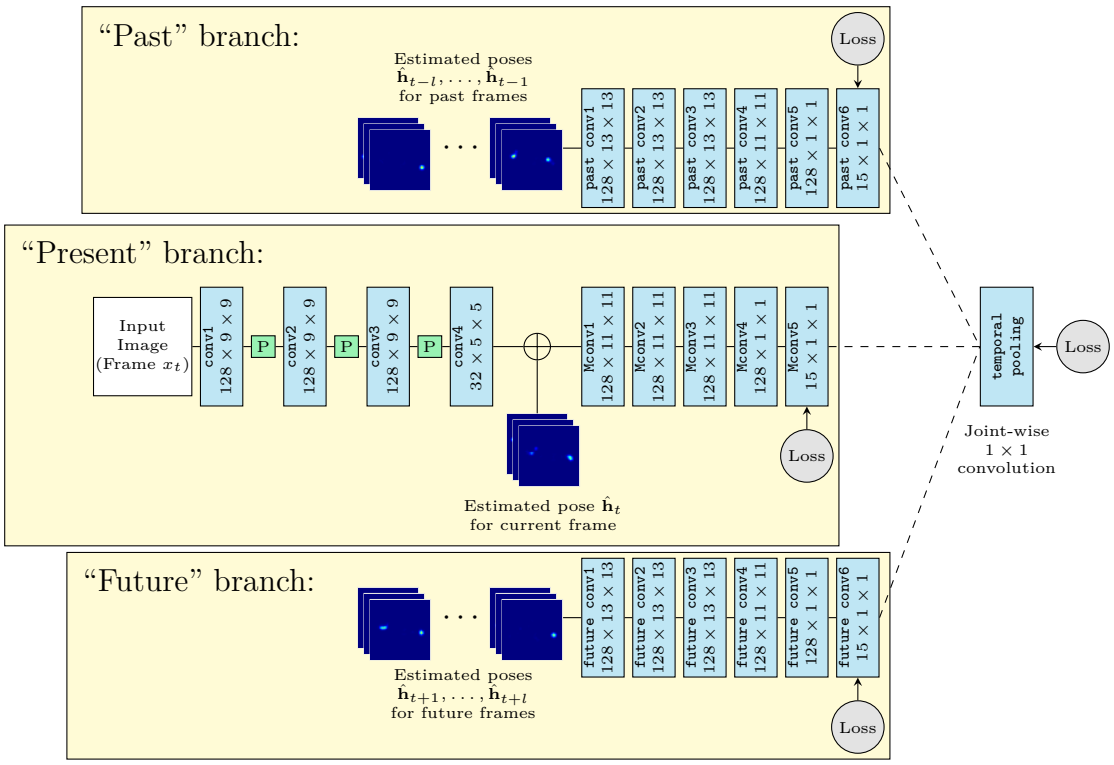


Figure 6.8: Network architecture for sequential pose refinement with temporal pooling. Pose estimates from past and future frames are processed in separate network branches. Each branch is trained to predict the pose for the current frame t. The temporal pooling layer combines predicted heatmaps from all three branches and is trained separately.

average to combine the prediction heatmaps for one specific joint. It consists of only three weights, one for each branch.

Training is again split up in multiple phases. All three branches are trained separately. The outer branches are trained from scratch for 70k iterations, the branch in the middle is as usual initialized using the stage-3 weights of the baseline swimmer CPM and then trained for 10k iterations. All branches use the common euclidean loss w.r.t. to the ground truth heatmaps. In a third step, the temporal pooling layer is trained exclusively. Since it only uses 15×3 weights to calculate a weighted average and no further backpropagation is performed, 400 iterations suffice for the loss to converge.

6.3.2 Evaluation

We show the result of the proposed architecture using temporal pooling in Figure 6.9. Compared to the previous version of the sequential post-processing stage, it is able to further improve pose estimation across all swimming styles. For backstroke,

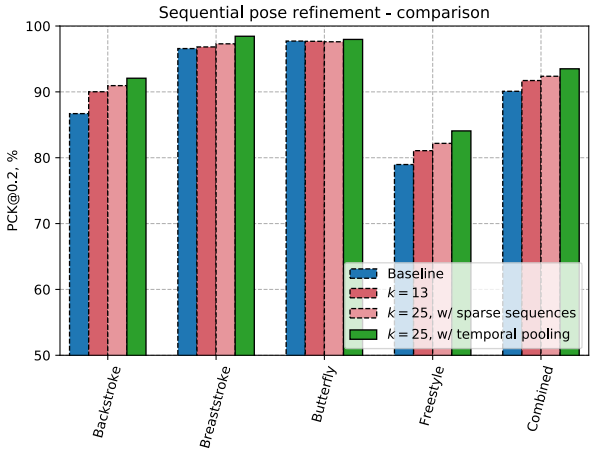


Figure 6.9: Results of the network architecture with temporal pooling. Performance from previous experiments is shown for comparison (stroked).

past conv6	Mconv5	future conv6
0.289	0.300	0.306

Table 6.1: Temporal pooling calculates a joint-wise weighted average on the output layers of the three network branches. The table shows the weights averaged over all joints.

breaststroke and butterfly, the PCK results of 92.1%, 98.4% and 98.0% respectively are among the best scores achieved throughout this work. Only the freestyle result of 84.1% can not compete with the CPM models using swimming style information in Section 5.1.

With the simple design of the temporal pooling layer, its weights can be analyzed to see what influence each network branch has on the final output. The layer consists of 3 weights (and one bias term) for each joint, which are used to compute a weighted average on the past, present and future prediction heatmaps. Table 6.1 depicts the layer weights averaged over all joints. Apparently, similar importance is assigned to all three branches. This shows that the provided sequential pose information has a notable influence on the final prediction. Combined, the predictions based on past and future poses could even outvote the single-frame prediction.

In contrast to the Flowing ConvNet, the network can also be trained jointly with training loss only calculated after the temporal pooling layer and then propagated backwards through all network branches. But it leads to a performance decrease for all swimming styles, up to -2% in PCK. Strictly enforcing each network branch to accurately predict the current frames pose proves to be the superior approach.

6.4 Conclusion

This chapter presented ideas on how to extend the single-frame CPM to human pose estimation on videos. The theoretical benefit of sequential video frames on pose estimation is apparent. But processing videos in (convolutional) neural networks is still an open question. Therefore, the presented network architectures do not operate on the sequential video frames themselves, but on the sequential estimates from the single-frame CPM. They are intended as a refinement mechanism to detect and correct possible errors in a pose sequence. It was observable that processing longer sequences can gradually improve performance. But simply providing the sequential information to the network does not suffice. The network has to be guided to utilize this information appropriately. The limited training data might be one reason, since the single-frame CPM already achieved a training set error close to zero.

The observations motivated a modified version of the post-processing CPM stage where information for the current and for past and future frames is processed by different parts of the network. Temporal pooling i used to combine the predictions of separate network branches. This architecture led to another increase in performance and partially surpassed the CPM models using swimming style information from previous sections. But where the swimming style models only need small additions to some of the CPM network layers, the sequential post-processing requires another convolutional network with considerable overhead for training and application. Efficiently utilizing sequential information stays a difficult challenge. The presented architectures for pose refinement were applied to the single-frame estimates of the baseline swimmer CPM to study the effect of sequential information alone. However, a combination with swimming style information is straightforward and might lead to further improvements.

7 Manual Network Activation Injection for Pose Correction

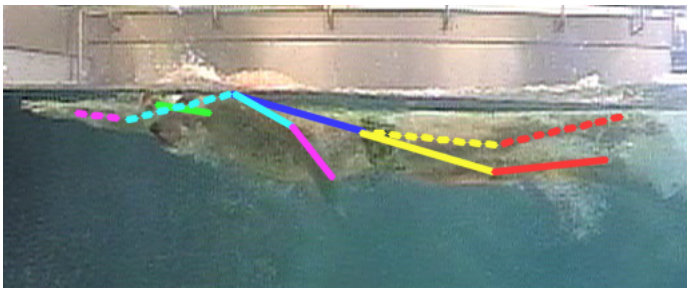
The major advantage of CPMs is the ability to implicitly learn dependencies between body parts without any form of supervision to enforce such behavior. However, there is no guarantee that the pose estimates resemble valid joint configurations. Each predicted joint location is obtained separately from its output heatmap as the location with highest confidence. But there are various methods for subsequent correction mechanisms to detect invalid configurations and improve pose estimates in general (see e.g. [PIT+16]). In this chapter we focus on difficult examples where the CPM alone performs poorly. We use an external mechanism for correction proposals and incorporate this additional information into the CPM pipeline.

For this purpose, we take a closer look at the network activations throughout the CPM architecture to analyze how erroneous predictions evolve over the stage-wise refinement process. This helps to identify possible locations in the architecture to inject corrections.

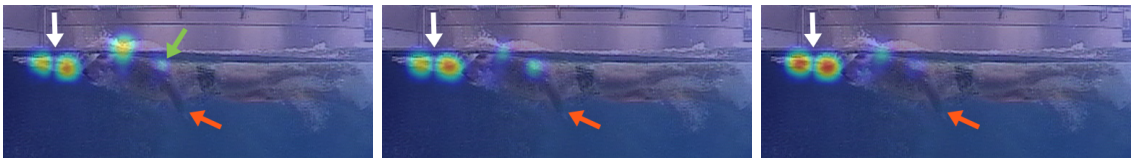
7.1 Network Activations on Failure Cases

The main idea of the CPM architecture is to first use local image evidence to find possible joint locations and then incorporate spatial context for refinement. Stage 1 is limited to local image information due to its receptive field. This usually leads to multiple hypotheses where a particular joint can be located due to similar appearance (e.g. wrists and ankles). Additionally, it is difficult to identify left and right joints from local information only. Subsequent stages can then refine these initial estimates by amplifying correct hypotheses and suppressing others.

In the case of swimming channel recordings, variety in poses and appearance is limited. When comparing the training loss of the baseline swimmer CPM, loss f_1 after stage 1 is already very small and in the same order of magnitude as the loss for subsequent stages. This means that during training, the output heatmaps of stage 1 are already close to the ground truth and do not necessarily contain multiple hypotheses. It remains to be seen whether this is also the case during evaluation on previously unseen test set examples.



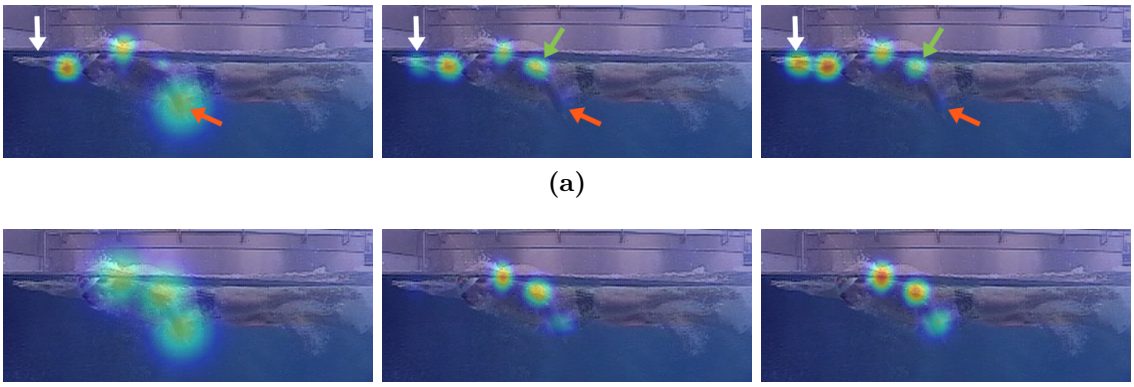
(a)



(b)

Figure 7.1: Estimated pose on a freestyle example. (a) Joints of the left and right arm are confused. (b) Output heatmaps for the left shoulder, elbow and wrist for stages 1–3 (left to right). Heatmaps for the three joints are combined into a single image for easier visualization. Wrong activations prevail (white arrow), missing activations on the left wrist after stage 1 are not regenerated by subsequent stages (orange arrow).

As an example, Figure 7.1a depicts the estimated pose on a freestyle test set frame using the baseline swimmer CPM. The joints of the left and right arm are mixed up – a frequent error on freestyle images. Figure 7.1b visualizes how the heatmaps for joints of the left arm evolve over the three stages. Note that the individual heatmaps for wrist, arm and shoulder of one stage are stacked into a single image for easier visualization. After the first stage, erroneous activations on the right elbow and wrist can be observed (white arrow). Only small (elbow, green arrow) or no activations at all (wrist, orange arrow) are present at the correct locations. On this example, stage 1 already tries to identify whether a joint belongs to the left or right side of the body, but fails in doing so. This differs to the stage 1 output of the CPM trained on the Leeds Sports Pose dataset (see Figure 2.4). It might be an artifact of the limited swimmer training data. The remaining stages are not able to suppress the wrong activations. Most prominently, the absence of activation on the left wrist prevails. This shows that the swimmer CPM might not be able to regenerate missing activations from stage 1. However, if an external mechanism provided a hint where the left wrist is roughly located, it could be possible to replenish the stage 1 activations in that area to direct the subsequent stages to the correct location.



(a)

(b)

Figure 7.2: Combined output heatmaps for the left shoulder, elbow and wrist for stages 1–3 (left to right) with stage 1 activation correction. (a) Only the left wrist is corrected. (b) All joints of the left arm are corrected.

7.2 Directing the Network to Correct Joint Locations

As a proof of concept to encode correction hints into stage 1 activations, we use ground truth information to identify wrong joints and inject corrected activations into the CPM. We inject these corrections into the output of stage 1 after its last layer `conv7` (see Figure 2.3 again for the default CPM architecture). For each wrongly localized joint, the stage 1 output heatmap is replaced by a heatmap with a single gaussian activation at the ground truth location. Then a second forward pass starting at stage 2 is performed to observe if the remaining stages are able to utilize the injected correction. In practice, the proposals from an external error detection and correction system can of course be erroneous as well. Therefore, the correction hints in stage 1 are encoded with additional uncertainty. We use a gaussian with large standard deviation and normalize its maximum to 0.5. Note that the subsequent stages of the baseline swimmer CPM are not trained on this altered input. The goal of this experiment is to observe if the network can cope with such previously unseen activations and utilize the correction hints. Since errors on swimmer examples often affect all joints of a limb, it is also interesting to see if the correction of single joints suffices or if connected joints have to be manually corrected as well.

7.2.1 Qualitative Example

As a first qualitative example, we test the activation correction on the previous freestyle image where the left arm is mistaken for the right arm and vice versa. Figure 7.2 visualizes the combined heatmaps of all three left-arm joints after stages 1-3. In Figure 7.2a, only the location of the left wrist is corrected. Note the wide

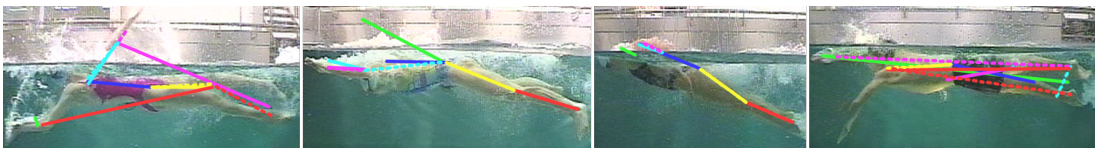


Figure 7.3: Failure cases on previously unseen videos included in the "hard examples" swimmer dataset. Backstroke, breaststroke, butterfly, freestyle – from left to right.

gaussian activation after stage 1 (orange arrow). Despite the activation being at the correct location, it is suppressed by the subsequent stages. The correction of a single joint does not suffice here. It is outweighed by related joint estimates. However, there are two noteworthy observations. The stage 2 heatmap for the left wrist again contains activations on the (wrong) right wrist, even though the corrected stage 1 heatmap contains no activation at this location (white arrow). This shows that later stages are still able to generate new hypotheses that are not present after the first stage. Secondly, while the left wrist is not detected, the activation on the left elbow (green arrow) increases compared to Figure 7.1b. Even if correction proposals do not cover all erroneous joints, the correction of some can improve the estimates of others. In Figure 7.2b, all joints of the left arm are corrected in the stage 1 heatmaps. This eliminates all ambiguities about the configuration of the left arm. The remaining stages now successfully preserve the corrected locations and just refine the spatial uncertainty of the injected gaussian activations. This is however a rather optimistic case since the corrections from an external mechanism will not always cover all wrong joints.

7.2.2 Quantitative Evaluation

For a quantitative evaluation, we apply the baseline swimmer CPM to previously unseen videos and pick 25 frames for each swimming style by hand that contain the most estimation errors. Figure 7.3 illustrates examples of these failure cases. Errors on breaststroke and butterfly examples are usually limited to few joints. For backstroke and freestyle, the estimates contain completely invalid joint configurations, with arms and legs mixed up and the head being misplaced. This smalls dataset of 100 hard examples is now used to evaluate how much correction is necessary to improve pose estimation to the performance level of previous chapters.

Figure 7.4a depicts the initial joint-wise PCK@0.2 scores on the hard examples using the baseline swimmer CPM. The overall PCK is at 70.1%, with backstroke and freestyle being as low as 57.1% and 55.4% respectively. We now apply the activation correction using ground truth information. In the first experiment, joints are corrected separately. For each wrong joint, its stage 1 heatmap is corrected, a forward pass through the network is performed and the new estimate for this joint is extracted from the stage 3 output heatmap. The experiment is intended

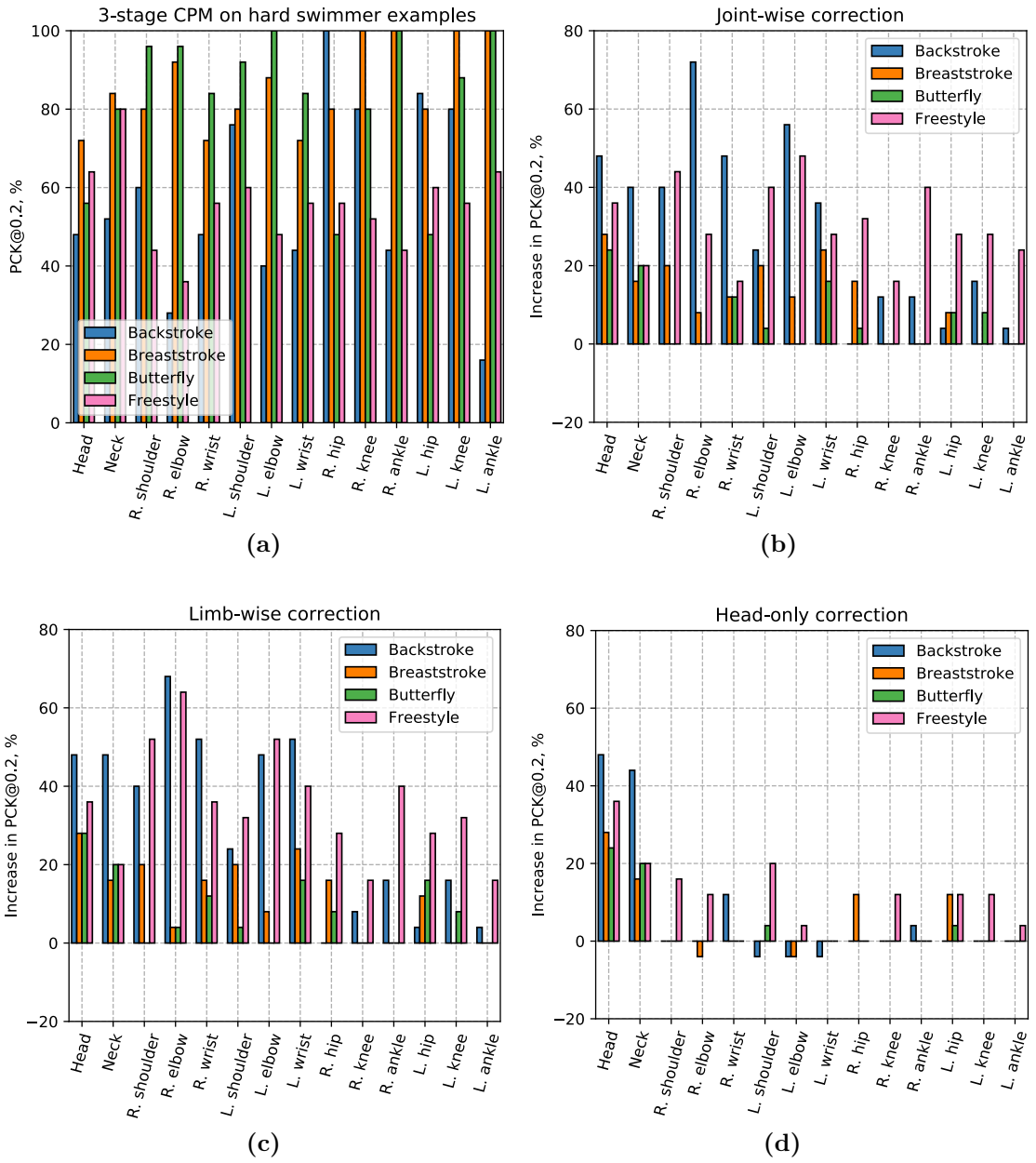


Figure 7.4: Joint-wise results on the "hard examples" swimmer dataset using ground-truth based correction. (a) Results of the baseline swimmer CPM. (b) Increase in PCK when stage 1 activations are corrected one joint at a time. (c) Increase in PCK with limb-wise correction. (d) Increase in PCK when only the head is corrected.

	Backstroke	Breaststroke	Butterfly	Freestyle	Combined
No correction	57.1	85.7	82.3	55.4	70.1
Joint-wise corr.	86.6	97.4	89.1	86.0	89.8
Limb-wise corr.	87.7	97.4	**90.6**	**90.6**	91.6
All-joint corr.	**91.4**	**98.3**	90.3	88.9	**92.2**
Head-only corr.	64.0	90.0	86.0	66.0	76.5

Table 7.1: PCK@0.2 results of ground truth based pose correction with the baseline swimmer CPM on the hard swimmer examples. Results are shown for no correction (baseline), joint-wise and limb-wise correction and correction of all joints simultaneously. The result when only the head is corrected is also shown.

to reveal if the network can already benefit from corrections for single joints only. Figure 7.4b depicts the increase in PCK performance compared to the baseline result. Surprisingly, the joint-wise corrections already leads to a vast improvement to an overall PCK of 89.8%. The improvements are not limited to joints like the head or neck that need less contextual information to be detected. Thus, in contrast to the example in Figure 7.2a, single-joint corrections alone can already provide important clues for the CPM network. Figure 7.4c shows the result of the second experiment where corrections are performed limb-wise, e.g. the left shoulder, elbow and wrist are corrected simultaneously. Providing corrections for connected joints at the same time leads to an additional increase of +1.8% PCK on the complete dataset. The improvements stem mainly from arm-joints in freestyle examples. A third experiment where all body joints are corrected simultaneously only reveals a small additional increase of +0.6%. The results on individual swimming styles for all experiments are summarized in Table 7.1.

Of course, the results in the experiments are overly optimistic since ground truth annotations are used to detect all wrong joints and inject stage 1 activations at the expected locations. Two further experiments are conducted for more realistic results. Figure 7.4d depicts the improvement for all joints when only the ground truth location of the head is used for correction. Even in this scenario, the overall performance can be improved by +6.4% compared to the baseline result. Since the detection of the head is crucial to determine the orientation of a swimmer, this single correction can improve other joints as well.

At last, we cannot expect the proposals from external pose correction to exactly match the ground truth. This situation is simulated in Figure 7.5, where all wrong joints are corrected simultaneously using noisy ground truth annotations: each annotated joint location is displaced in a random direction by a random PCK distance in $[0, \alpha]$. We observe that even with $\alpha = 0.4$, i.e. on average half the corrections being wrong w.r.t. the PCK@0.2 definition, an overall performance of 87.6% is achieved. This is only 4.6% lower compared to correction without ground truth noise. This shows that the network can handle possibly wrong correction hints and thus makes it applicable to proposals from a real correction system.

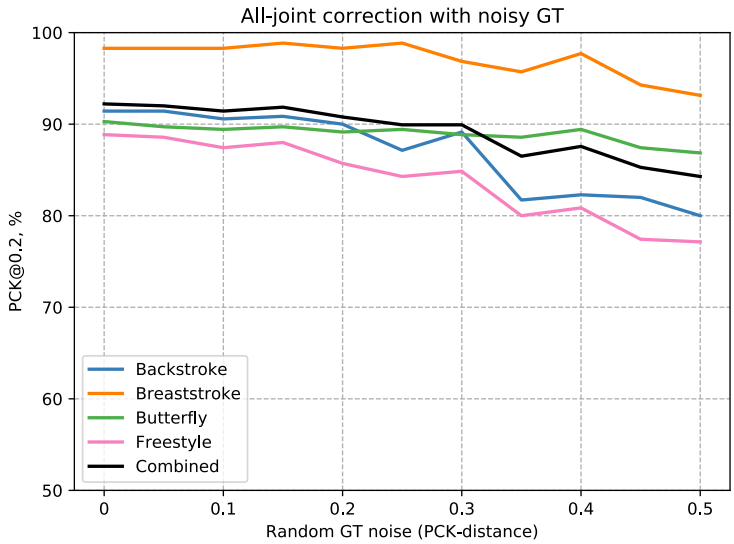


Figure 7.5: Results on the "hard example" swimmer dataset with simultaneous correction of all wrong joints. Ground truth locations used for correction are randomly shifted to simulate noisy corrections. Noise offset is drawn from $[0, \alpha]$ PCK-distance.

7.3 Using Proposals from Sequence-Based Joint Correction

We now use an external error detection and correction system to proof that the concept of activation correction also works in a practical application. The correction system is based on the observation that cyclic swimming motion in videos leads to a periodic pattern regarding the x and y image coordinates of joints. It uses the predicted joint coordinates over a fixed-length sequence of video frame and approximates the coordinates over time with a polynomial function. It then identifies outliers that deviate from the approximated function and corrects them. We use this correction system as a closed black-box that provides corrected locations for all joints in a video. Note that the internal parameters of the system are optimized for freestyle movement.

7.3.1 Evaluation

In order to use the correction system on the hard swimmer examples, we provide joint predictions for the complete swimmer videos the failure examples were picked from. This is necessary since the correction system operates on predictions from sequential video frames. The resulting corrected predictions are then filtered for the frames of interest.

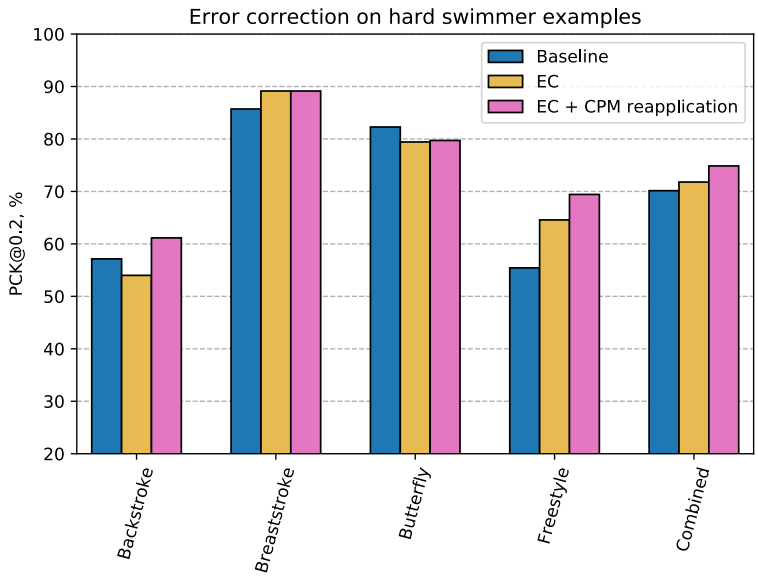


Figure 7.6: Results on the "hard example" swimmer dataset using external correction (EC) proposals. Scores are shown for the baseline swimmer CPM without correction, the external correction mechanism itself and the reapplication of corrections to CPM stage 1.

Figure 7.6 depicts the result when the output of the external correction (EC) system is evaluated itself. Compared to the initial baseline swimmer predictions it slightly improves PCK@0.2 to 71.8% on the complete dataset. The largest gain of +9.2% is observed for the freestyle examples. This is to be expected since the correction system is optimized on freestyle videos. But the decline in performance for backstroke and butterfly shows that there is no guarantee that the corrections are actually better than the initial CPM predictions.

We now use the correction proposals and reapply them to the stage 1 activations of the baseline swimmer CPM. The correction system possibly corrects all joints, even if only by very small amounts. However, we want to preserve as much of the original stage 1 activations as possible and only change those heatmaps where the corrected joint location clearly differs from the original prediction. It is therefore necessary to only apply joint corrections that have at least 0.3 PCK-distance to the initial prediction.

With this setting, the joint corrections are added to the stage 1 activations, a second forward pass is performed and the new network output is used to obtain refined corrections for all joints. We show the result of this additional refinement (denoted EC + CPM reapplication) in Figure 7.6. Combined performance of all swimming styles further improves to 74.9%. Performance on the symmetrical swimming styles (breaststroke, butterfly) is nearly identical to the EC result. For the freestyle corrections, the CPM refinement adds another +4.8% PCK. Interestingly, the refinement increases backstroke performance above the baseline, despite the inferior result of

the external correction system alone. This indicates that our approach of CPM refinement also works in the case of erroneous corrections.

7.4 Conclusion

In this chapter we presented idea to combine single-frame CPM predictions and an external system for pose correction. We developed a method to reintegrate correction hints about possible joint locations into the CPM by artificially adding network activations in the intermediate heatmaps after stage 1. Experiments using ground-truth based correction showed that even correction hints for only some of the wrongly predicted joints can help to increase overall performance. In order to verify the findings in a realistic setting, we used joint corrections based on polynomial approximation of joint coordinates over time. Reapplying the corrections to the CPM led to an additional increase in PCK performance on difficult swimmer examples. The approach resembles an efficient way to use sequential information for human pose estimation: Single-frame CPM estimates and interpolation of discrete joint locations over time. Furthermore, the manual addition of network activations is not limited to posterior pose correction but could also be used to encode a prior belief of where joints are probably located.

8 Summary and Future Work

8.1 Summary

The objective of this thesis was to understand the concept of Convolutional Pose Machines, extend it with additional input and secondary learning objectives and apply it to a real-world scenario. We described the CPM framework and its stage-wise network architecture and reproduced published benchmark results on the popular Leeds Sport Pose dataset. The results motivated the further usage of a 3-stage architecture to reduce the memory footprint and keep training and evaluation time feasible.

Since occluded body parts and joints are a common challenge in human pose estimation, we analyzed the confidence output on these joints. We argued that as a baseline approach, low confidence scores can be used as an indicator for occlusion. We developed a first CPM extension to explicitly predict joint visibility, which clearly outperformed the baseline result. It showed that additional objectives can be added rather easily to the network architecture.

We successfully applied CPMs to video recordings from a swimming channel environment despite its unique challenges. The swimmer videos were used as a benchmark throughout this work. Because performance depended heavily on the swimming style, we discussed how this contextual information can be utilized by a CPM. For this purpose, we developed a generic CPM extension to input class label information into a fully convolutional architecture. It resulted in a notable improvement on the swimmer dataset. This extension is not limited to swimming styles but could for example also encode different sports or activities in general. We also showed that the swimming style can be inferred reliably by a CPM, under the assumption of video frames being processed sequentially.

For humans, the subjective benefit of sequential information in videos is huge. Estimating movement over time can be essential for joints that are otherwise difficulty to detect in single video frames. This motivated the extension of the CPM framework from single-frame operation to videos. We presented a separate post-processing network to refine the estimates of sequential video frames. An extended architecture using the concept of temporal pooling achieved promising results. But it incurs a considerable overhead in computation.

Finally, we discussed how a CPM can be combined with a system that detects and roughly corrects erroneous joint predictions. This lead to a generic method to

encode additional belief about joint locations into the CPM network activations.

In general, the CPM architecture presented itself as quite flexible. Training the presented extensions did rarely require careful adjustment of learning parameters.

8.2 Future Work

The concept of CPMs is already included in other architectures. For example, Cao et al. transfer CPMs to multi-person pose estimation [CSWS17]. We intend to use a CPM in a productive system for pose estimation in the swimming channel scenario. We still believe that sequential video frames possess benefits that are yet to be exploited. We will therefore try to not only post-process sequential estimates but to directly use image information from subsequent frames. Adding a recurrent mechanism [BZ17] or the prediction-correction building blocks proposed in [DRR17] both seem to be promising approaches. It will be interesting to see how videos can be processed efficiently in convolutional neural networks.

Bibliography

[APGS14] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. (Cited on page 10.)

[ARS09] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1014–1021, June 2009. (Cited on pages 3 and 5.)

[BZ17] Vasileios Belagiannis and Andrew Zisserman. Recurrent human pose estimation. *arXiv e-prints*, February 2017. URL: https://arxiv.org/abs/1605.02914v2. (Cited on page 82.)

[Car96] Rich Caruana. Algorithms and applications for multitask learning. In *ICML*, pages 87–95, 1996. (Cited on page 46.)

[CSWS17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. *arXiv e-prints*, April 2017. URL: https://arxiv.org/abs/1611.08050. (Cited on page 82.)

[DG06] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006. (Cited on pages 20 and 49.)

[DRR17] Achal Dave, Olga Russakovsky, and Deva Ramanan. Predictive-Corrective Networks for Action Detection. *arXiv e-prints*, April 2017. URL: https://arxiv.org/abs/1704.03615. (Cited on page 82.)

[FGH13] Mykyta Fastovets, Jean-Yves Guillemaut, and Adrian Hilton. Athlete pose estimation from monocular tv sports footage. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013. (Cited on page 29.)

[FH05] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1): 55–79, 2005. (Cited on page 5.)

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. (Cited on pages 25, 31, 47, 57 and 58.)

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. (Cited on pages 5 and 36.)

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. (Cited on page 36.)

[GM13] Rikke Gade and Thomas B. Moeslund. Sports type classification using signature heatmaps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013. (Cited on pages 29 and 47.)

[HS16] Kunihiro Hasegawa and Hideo Saito. Synthesis of a stroboscopic image from a hand-held camera sequence for a sports analysis. *Computational Visual Media*, 2(3): 277–289, 2016. (Cited on page 29.)

[Ins17] Max Planck Institut. MPII human pose dataset, URL: `http://human-pose.mpi-inf.mpg.de`. (Cited on page 13.)

[IPA+16] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016. (Cited on page 1.)

[JE10] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*, 2010. (Cited on page 10.)

[JE11] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011. (Cited on page 10.)

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. (Cited on page 11.)

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. (Cited on pages 5 and 47.)

[LAE+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016. (Cited on page 36.)

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, Nov 1998. (Cited on page 47.)

[LH05] Xiangyang Lan and Daniel P Huttenlocher. Beyond trees: Common-factor models for 2D human pose recovery. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 470–477. IEEE, 2005. (Cited on page 5.)

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. (Cited on page 5.)

[MTH15] Thomas B. Moeslund, Graham Thomas, and Adrian Hilton. *Computer vision in sports*. Springer, 2015. (Cited on page 29.)

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016. (Cited on page 1.)

[PCZ15] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing Conv–Nets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1913–1921, 2015. (Cited on pages 6, 58 and 66.)

[PIT+16] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, and Bernt Schiele. DeepCut: Joint subset partition and labeling for multi person pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. (Cited on page 71.)

[PSEAG14] Xavier Perez-Sala, Sergio Escalera, Cecilio Angulo, and Jordi Gonzàlez. A survey on model based approaches for 2D and 3D visual human pose recovery. *Sensors*, 14(3): 4189–4210, 2014. (Cited on page 5.)

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. (Cited on page 36.)

[RMH+14] Varun Ramakrishna, Daniel Munoz, Martial Hebert, James Andrew Bagnell, and Yaser Sheikh. Pose machines: Articulated pose estimation via inference machines. In *European Conference on Computer Vision*, pages 33–47. Springer, 2014. (Cited on page 6.)

[SKN10] V. K. Singh, F. M. Khan, and R. Nevatia. Multiple pose context trees for estimating human pose in object context. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition - Workshops*, pages 17–24, June 2010. (Cited on page 39.)

[SKS15] Khurram Soomro, Salman Khokhar, and Mubarak Shah. Tracking when the camera looks away. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015. (Cited on page 29.)

[SSK+13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1): 116–124, January 2013. (Cited on page 17.)

[ST13] Ben Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3681, 2013. (Cited on pages 5, 10 and 12.)

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv e-prints*, April 2015. URL: https://arxiv.org/abs/1409.1556. (Cited on page 11.)

[TDB16] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3D models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016. (Cited on page 40.)

[TS14] Alexander Toshev and Christian 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. (Cited on pages 5, 6, 8, 13, 21 and 26.)

[TSDB15] Francesco Turchini, Lorenzo Seidenari, and Alberto Del Bimbo. Understanding sport activities from correspondences of clustered trajectories. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015. (Cited on pages 29 and 47.)

[WRKS16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016. (Cited on pages 1, 5, 6, 9, 10, 11, 12, 13, 17, 18 and 60.)

[WSL+15] Xinyu Wei, Long Sha, Patrick Lucey, Peter Carr, Sridha Sridharan, and Iain Matthews. Predicting ball ownership in basketball from a monocular view using only player trajectories. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015. (Cited on page 29.)

[YR13] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12): 2878–2890, 2013. (Cited on page 12.)

[ZL15] Dan Zecha and Rainer Lienhart. Key-pose prediction in cyclic human motion. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 86–93. IEEE, 2015. (Cited on page 30.)

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my reviewer Prof. Dr. Rainer Lienhart for the possibility to write my master thesis at the Multimedia Computing and Computer Vision Lab under his guidance and for all the provided resources.

Secondly, I am very grateful to Dan Zecha for his supervision, the collaboration and all his advice for my start into the scientific world.

Many thanks to my family in total for their support throughout the last years. Special thanks in particular to Alexandra Feß for her encouragement during the last months.