Chapter 4

Pose Representation using Human Pose Estimation Algorithms

4.1 Introduction

In this chapter, we describe our first pose representation. This pose representation is derived from the output of human pose estimation algorithms (HPE). The HPE task is to predict the 2D (image) stickman layout of the person: head, torso, upper and lower arms. While HPE algorithms [26, 80, 5, 103] have a reasonable success, it suffers from unreliability. With good frequency, a few body parts of the person (typically lower arms) are incorrectly estimated. This clearly affects the human pose retrieval task. In this chapter, we address the issue of reliability and describe the procedure to obtain the compact 12D pose representation from the output of the human pose estimation algorithm. This work [50] is published in ECCV 2012.

Computer vision algorithms for recognition are getting progressively more complex as they build on earlier work including feature detectors, feature descriptors and classifiers. A typical object detector, for example, may involve multiple features and multiple stages of classifiers [33]. However, apart from the desired output (e.g. a detection window), at the end of this advanced multi-stage system the sole indication of how well the algorithm has performed is typically just the score of the final classifier [35, 98]. In this chapter, we describe how to redress this balance and obtain pose representations. We argue that algorithms should and can self-evaluate (illustrated in figure 4.1). They should self-evaluate because this is a necessary requirement for any practical systems to be reliable. That they can self-evaluate is demonstrated in this work for the case of human pose estimation (HPE). Such HPE evaluators can then take their place in the standard armoury of many applications, for example removing incorrect pose estimates in video surveillance, pose based image retrieval, or action recognition.

In general four elements are needed to learn an evaluator: a ground truth annotated database that can be used to assess the algorithm’s output; a quality measure comparing the output to the ground truth; auxiliary features for measuring the output; and a classifier that is learnt as the evaluator. After learning, the evaluator can be used to predict if the algorithm has succeeded on new test data, for which ground-truth is not available. We discuss each of these elements in turn in the context of HPE.
The ground truth annotation in the dataset must provide the positions of various body parts. The pose quality is measured by the difference between the predicted layout and ground truth – for example the difference in their angles or joint positions. The auxiliary features can be of two types: those that are used or computed by the HPE algorithm, for example max-marginals of limb positions; and those that have not been considered by the algorithm, such as proximity to the image boundary (the boundary is often responsible for erroneous estimations). The estimate given by HPE on each instance from the dataset is then classified, using a threshold on the pose quality measure, to determine positive and negative outputs (e.g. based on the number of correctly estimated limbs). Given these positive and negative training examples and both types of auxiliary features, the evaluator is learnt using standard methods (here an SVM).

We apply this evaluator learning framework to four recent publicly available methods: Eichner and Ferrari [26], Sapp et al. [80], Andriluka et al. [5] and Yang and Ramanan [103]. The auxiliary features, pose quality measure, and learning method are described in section 4.2. For the datasets, section 4.3, we use existing ground truth annotated datasets, such as ETHZ PASCAL Stickmen [26] and Humans in
3D [13], and supplement these with additional annotation where necessary. We assess the evaluator features and method on the four HPE algorithms, and demonstrate experimentally that the proposed evaluator can reliably predict when the algorithms succeed. In this chapter, we extend our preliminary work [50] on this subject with new ways in which pose evaluator can be used to filter out incorrect poses produced by a pose estimator and to combine the outputs of multiple different estimators (section 4.4).

Note how the task of a HPE evaluator is not the same as that of deciding whether the underlying human detection is correct or not. It might well be that a correct detection then leads to an incorrect pose estimate. Moreover, the evaluator cannot be used directly as a pose estimator either – a pose estimator predicts a pose from an enormous space of possible structured outputs. The evaluator’s job of deciding whether a pose is correct is different, and easier, than that of producing a correct pose estimate.

On the theme of evaluating vision algorithms, the most related work to ours is Mac Aodha et al. [64] where the goal is to choose which optical flow algorithm to apply to a given video sequence. They cast the problem as a multi-way classification. In visual biometrics, there has also been extensive work on assessor algorithms which predict an algorithm’s failure [81, 99, 1]. These assess face recognition algorithms by analyzing the similarity scores between a test sample and all training images. The method by [99] also takes advantage of the similarity within template images. But none of these explicitly consider other factors like imaging conditions of the test query (as we do). [1] on the other hand, only takes the imaging conditions into account. Our method is designed for another task, HPE, and considers several indicators all at the same time, such as the marginal distribution over the possible pose configurations for an image, imaging conditions, and the spatial arrangement of multiple detection windows in the same image. Other methods [101, 12] predict the performance by statistical analysis of the training and test data. However, such methods cannot be used to predict the performance on individual samples, which is the goal of this work.

### 4.2 Pose Evaluation Method

We formulate the problem of evaluating the human pose estimates as classification into ‘success’ and ‘failure’ classes. First, we describe the novel features we use (section 4.2.1). We then explain how human pose estimates are evaluated. For this, we introduce a measure of quality for HPE by comparing it to a ground-truth stickman (section 4.2.2). The features and quality measures are then used to train the evaluator as a classifier (section 4.2.3).

#### 4.2.1 Features

We propose a set of features to capture the conditions under which an HPE algorithm is likely to make mistakes. We identify two types of features: (i) based on the output of the HPE algorithm – the score of the HPE algorithm, marginal distribution, and best configuration of parts $L^*$; and, (ii) based
Figure 4.2: Features based on the output of HPE. Examples of unimodal, multimodal and large spread pose estimates. Each figure (first row) is overlaid with the best configuration (sticks) and the posterior marginal distribution over the body part position in a semi-transparent mask (displayed in second row). ‘Max’ and ‘Var’ features are measured from this distribution (third row). In the case of unimodal distributions, as the above examples (a and b) indicate, the mode almost always corresponds to the correct part location. While in the case of multimodal distributions, typically one of the modes corresponds to the correct part location as in example (c), but again they may not as in example (d). Finally in the case of diffuse distributions (e and f), the many modes that barely stand out do not convey any information about correctness of part locations. Thus unimodal distributions are good indicators of correct part configurations as opposed to the other two types. The type of distribution is determined using the ‘Max’ and ‘Var’ features. As the distribution moves from peaked unimodal to more multi-modal and diffuse, the maximum value decreases and the variance increases.

1. Features from the output of the HPE. The outputs of the HPE algorithm consist of a marginal probability distribution $P_i$ over $(x, y, \theta)$ for each body part $i$ and the best configuration of parts $L^*$. The features computed from these outputs measure the spread and multi-modality of the marginal distribution of each part. As can be seen from figures 4.2 and 4.3, the multi-modality and spread correlate well with the error in the pose estimate. Features are computed for each body part in two stages: first, the marginal distribution is pose-normalized, then the spread of the distribution is measured by comparing it to an ideal signal.

The best configuration $L^*$, predicts an orientation $\theta$ for each part. This orientation is used to pose-normalize the marginal distribution (which is originally axis-aligned) by rotating the distribution so that its orientation agrees with $\theta$.
that the predicted orientation corresponds to the $x$-axis (illustrated in figure 4.4). The pose-normalized marginal distribution is then factored into three separate $x$, $y$ and $\theta$ spaces by projecting the distribution onto the respective axes. Empirically we have found that this step improves the discriminability.

A descriptor is then computed for each of the separate distributions which measure its spread. For this we appeal to the idea of a matched filter, and compare the distribution to an ideal unimodal one, $P^*$, which models the marginal distribution of a perfect pose estimate. The distribution $P^*$ is assumed to be Gaussian and its variance is estimated from training samples with near perfect pose estimate. The unimodal distribution shown in figure 4.2 is an example corresponding to a near perfect pose estimate.

The actual feature is obtained by convolving the ideal distribution, $P^*$, with the measured distribution (after the normalization step above), and recording the maximum value and variance of the convolution. Thus for each part we have six features, two for each dimension, resulting in a total of 36 features for an upper-body. The entropy and variance of the distribution $P_i$, and the score of the HPE algorithm are also used as features taking the final total to $36 + 13$ features.

**Algorithm specific details:** While the procedure to compute the feature vector is essentially the same for all four HPE algorithms, the exact details vary slightly. For Andriluka et al. [5] and Eichner and Ferrari [26], we use the posterior marginal distribution to compute the features. While for Sapp et al. [80] and Yang and Ramanan [103], we use the max-marginals. Further, in [103] the pose-normalization is omitted as max-marginal distributions are over the mid and end points of the limb rather than over the limb itself. For [103], we compute the max-marginals ourselves as they are not available directly from the implementation.

2. **Features from the detection window.** We now detail the 10 features computed over the extended detection window. These consist of the scale of the extended detection window, and the confidence score of the detection window as returned by the upper body detector. The remainder are:

- **Two image features:** the mean image intensity and mean gradient strength over the extended detection window. These are aiming at capturing the lighting conditions and the amount of background clutter. Typically HPE algorithms fail when either of them has a very large or a very small value.

- **Four location features:** these are the fraction of the area outside each of the four image borders. Algorithms also tend to fail when people are very small in the image, which is captured by the scale of the extended detection window.

The location features are based on the idea that the larger the portion of the extended detection window which lies outside the image, the less likely it is that HPE algorithms will succeed (as this indicates that some body parts are not visible in the image).

- **Two overlap features:** (i) the maximum overlap, and (ii) the sum of overlaps, over all the neighbouring detection windows normalized by the area of the current detection window. As illustrated in Figure 4.5 HPE algorithms can be affected by the neighbouring people. The overlap features capture the extent of occlusion by the neighbouring detection windows. Overlap with other people
indicates how close the they are. While large overlaps occlude many parts in the upper body, small overlaps also affect HPE performance as the algorithm could pick the parts (especially arms) from their neighbours.

While measuring the overlap, we consider only those neighbouring detections which have similar scale (between 0.75 and 1.25 times). Other neighbouring detections which are at a different scale typically do not affect the pose estimation algorithm.

### 4.2.2 Pose Quality Measure

For evaluating the quality of a pose estimate, we devise a measure which we term the *Continuous Pose Cumulative error* (CPC) for measuring the dissimilarity between two poses. It ranges in $[0, 1]$, with 0 indicating a perfect pose match. In brief, CPC depends on the sum of normalized distances between the corresponding end points of the parts. Figure 4.6 gives examples of actual CPC values of poses as they move away from a reference pose. The CPC measure is similar to the *percentage of correctly estimated body parts* (PCP) measure of [26]. However, CPC adds all distances between parts, whereas PCP counts the number of parts whose distance is below a threshold. Thus PCP takes integer values in $\{0, \ldots, 6\}$. In contrast, for proper learning in our application we require a continuous measure, hence the need for the new CPC measure.
Figure 4.4: Pose normalizing the marginal distribution. Marginal distribution (contour diagram in green) on the left is pose normalized by rotating it by an angle $\theta$ of the body part (line segment in grey) to obtain the transformed distribution on the right. In the inset, the pertinent body part (line segment in grey) is displayed as a constituent of the upper body.

In detail, the CPC measure computes the dissimilarity between two poses as the sum of the differences in the position and orientation over all parts. Each pose is described by $N$ parts and each part $p$ in a pose $A$ is represented by a line segment going from point $s_a^p$ to point $e_a^p$, and similarly for pose $B$. All the coordinates are normalized with respect to the detection window in which the pose is estimated. The angle subtended by $p$ is $\theta_a^p$. With these definitions, the $CPC(A, B)$ between two poses $A$ and $B$ is:

$$CPC(A, B) = \sigma \left( \sum_{p=1}^{N} w_p \left( \frac{||s_a^p - s_b^p|| + ||e_a^p - e_b^p||}{2 ||s_a^p - e_a^p||} \right) \right)$$

with $w_p = 1 + \sin \left( \frac{|\theta_a^p - \theta_b^p|}{2} \right)$

where $\sigma$ is the sigmoid function and $\theta_a^p - \theta_b^p$ is the relative angle between part $p$ in pose $A$ and $B$, and lies in $[-\pi, \pi]$. The weight $w_p$ is a penalty for two corresponding parts of $A$ and $B$ not being in the same direction. Figure 4.7 depicts the notation in the above equation.

### 4.2.3 Learning the Pose Quality Evaluator

We require positive and negative pose examples in order to train the evaluator for an HPE algorithm. Here a positive is where the HPE has succeeded and a negative where it has not. Given a training dataset of images annotated with stickmen indicating the ground truth of each pose, (section 4.3.1), the positive and negative examples are obtained by comparing the output of the HPE algorithms to the ground truth using CPC and thresholding its value. Estimates with low CPC (i.e. estimates close to the true pose) are the positives, and those above threshold are the negatives.

In detail, the UBD algorithm is applied to all training images, and the four HPE algorithms [5, 26, 80, 103] are applied to each detection window to obtain a pose estimate. The quality of the pose estimate
is then measured by comparing it to ground truth using CPC. Analog to the use of PCP [26], CPC is only computed for correctly localized detections (those with IoU > 0.5). Detections not associated with any ground-truth are discarded. Since all of the HPE algorithms considered here cannot estimate partial occlusion of limbs, CPC is set to 0.5 if the ground-truth stickman has occluded parts. In effect, the people who are partially visible in the image get high CPC. This is a means of providing training data to the evaluator showing the HPE algorithms will fail on these cases.

A CPC threshold of 0.3 is used to separate all poses into a positive set (deemed correct) and a negative set (deemed incorrect). This threshold is chosen because pose estimates with CPC below 0.3 are nearly perfect and roughly correspond to PCP = 6 with a threshold of 0.5 (figure 4.6). A linear SVM is then trained on the positive and negative sets using the auxiliary features described in section 4.2.1. The feature vector has 59 dimensions, and is a concatenation of the two types of features (49 components based on the output of the HPE, and 10 based on the extended detection window). This classifier is the evaluator, and will be used to predict the quality of pose estimates on novel test images. The performance of the evaluator algorithm is discussed in the experiments of section 4.3.3.
Figure 4.6: **Poses with increasing CPC.** An example per CPC is shown for each of the two reference poses for CPC measures of 0.1, 0.2, 0.3 and 0.5. As can be seen, example poses move smoothly away from the reference pose with increasing CPC, with the number of parts which differ and the extent increasing. For 0.1 there is almost no difference between the examples and the reference pose. At 0.2, the examples and reference can differ slightly in the angle of one or two limbs, but from 0.3 on there can be substantial differences with poses differing entirely by 0.5.

### 4.3 Pose Evaluator Performance

After describing the datasets we experiment on (section 4.3.1), we present a quantitative analysis of the pose quality evaluator (section 4.3.3).

#### 4.3.1 Data

We experiment on four of the datasets introduced in chapter 3: *ETHZ PASCAL Stickmen* [26], *Humans in 3D* [13], *Buffy2 Stickmen* and *Movie Stickmen*. All four datasets are challenging as they show people at a range of scales, wearing a variety of clothes, in diverse poses, lighting conditions and scene backgrounds. Table 4.1 gives the number of images and annotated stickmen in each dataset. As a training set for our pose quality evaluator, we take Buffy2 Stickmen, Humans in 3D and five movies (*About a Boy, Apollo 13, Four Weddings and a Funeral, Forrest Gump, Notting Hill*) of Movie Stickmen. ETHZ PASCAL Stickmen and five movies (*Witness, Gandhi, Love Actually, The graduate, Groundhog day*) of
Figure 4.7: **Pose notation:** For poses A and B corresponding parts \( p \) are illustrated. Each part is described by starting point \( s_a^p \), end point \( e_a^p \) and the angle of the part \( \theta_a^p \) are also illustrated.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#images</th>
<th>#ground truth-train</th>
<th>#ground truth-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETHZ Pascal [27]</td>
<td>549</td>
<td>0</td>
<td>549</td>
</tr>
<tr>
<td>Humans in 3D [13]</td>
<td>429</td>
<td>1002</td>
<td>0</td>
</tr>
<tr>
<td>Movie [50]</td>
<td>5984</td>
<td>5804</td>
<td>5835</td>
</tr>
<tr>
<td>Buffy2 [50]</td>
<td>499</td>
<td>775</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>7270</td>
<td>7581</td>
<td>6834</td>
</tr>
</tbody>
</table>

Table 4.1: **Train-test split.** For our experiments, two publicly available datasets and two newly introduced datasets are used. The train-test stickmen annotations along with other statistics are given above.

Movie Stickmen form the *test set*, on which we report the performance of the pose quality evaluator in the next subsection.

### 4.3.2 Performance of the HPE Algorithms.

For reference, table 4.2 gives the average PCP across different datasets. Table 4.3 gives the percentage of samples where pose is estimated accurately, i.e. the CPC between the estimated and ground-truth stickmen is \(< 0.3\). In all cases, we use the implementations provided by the authors [26, 80, 5, 103] and all methods are given the same detection windows [32] as preprocessing. Both measures agree on the relative ranking of the methods: Yang and Ramanan [103] performs best, followed by Sapp *et al.* [80], Eichner and Ferrari [26] and then by Andriluka *et al.* [5]. This confirms experiments reported independently in previous works [26, 80, 103]. Note that we report these results only as a reference, as the absolute performance of the HPE algorithms is not important in this work. What matters is how well our newly proposed evaluator can predict whether an HPE algorithm has succeeded.
Datasets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffy stickmen</td>
<td>78.3</td>
<td>81.6 (83.3)</td>
<td>84.2</td>
<td>86.7</td>
</tr>
<tr>
<td>ETHZ Pascal</td>
<td>65.9</td>
<td>68.5</td>
<td>71.4 (78.2)</td>
<td>72.4</td>
</tr>
<tr>
<td>Humans in 3D</td>
<td>70.3</td>
<td>71.3</td>
<td>75.3</td>
<td>77.8</td>
</tr>
<tr>
<td>Movie</td>
<td>64.6</td>
<td>70.4</td>
<td>70.6</td>
<td>76.0</td>
</tr>
<tr>
<td>Buffy2</td>
<td>65.0</td>
<td>67.5</td>
<td>74.2</td>
<td>81.7</td>
</tr>
</tbody>
</table>

Table 4.2: Pose estimation evaluation (PCP). The PCP of the four HPE algorithms (Andriluka et al. [5], Eichner and Ferrari [26], Sapp et al. [80], Yang and Ramanan [103]) averaged over each dataset (at PCP threshold 0.5, section 4.2.2). The numbers in brackets are the results reported in the original publications. The differences in the PCPs are due to different versions of UBD software used. The performances across the datasets indicate their relative difficulty.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>8.5</td>
<td>10.7</td>
<td>11.6</td>
<td>18.5</td>
</tr>
<tr>
<td>Test</td>
<td>9.9</td>
<td>11.1</td>
<td>12.0</td>
<td>16.5</td>
</tr>
<tr>
<td>Total</td>
<td>9.1</td>
<td>10.9</td>
<td>11.8</td>
<td>17.6</td>
</tr>
</tbody>
</table>

Table 4.3: Pose estimation evaluation (CPC). The table shows the percentage of samples which were estimated accurately (CPC < 0.3) on the training and test sets, as well as overall, for the four HPE algorithms (Andriluka et al. [5], Eichner and Ferrari [26], Sapp et al. [80], Yang and Ramanan [103]). These accurate pose estimates form the positive samples for training and testing the evaluator.

### 4.3.3 Assessment of the Pose Quality Evaluator

Here we evaluate the performance of the pose quality evaluator for the four HPE algorithms. To assess the evaluator, we use the following definition: A pose estimate is defined as positive if it is within CPC 0.3 of the ground truth and as negative otherwise. The evaluator’s output (positive or negative pose estimate) is defined as successful if it correctly predicts a positive (true positive) or negative (true negative) pose estimate, and defined as a failure when it incorrectly predicts a positive (false positive) or negative (false negative) pose estimates. Using these definitions, we assess the performance of the evaluator by plotting an ROC curve.

The performance is evaluated under two regimes: (A) only where the predicted HPE corresponds to one of the annotations. Since the images are fairly completely annotated, any upper body detection window [32] which does not correspond to an annotation is considered a false positive. In this regime such false positives are ignored; (B) all predictions are evaluated, including false-positives. The first regime corresponds to: given there is a human in the image at this location, how well can the proposed method evaluate the pose estimate? The second regime corresponds to: given there are wrong person detections, how well can the proposed method evaluate the pose estimate? The protocol for assigning a HPE prediction to a ground truth annotation was described in section 4.2.3. For regime B, any pose on a false-positive detection is assigned a CPC of 1.0.
Table 4.4: Performance of the Pose Evaluator in Regime B. The pose evaluator is used to assess the outputs of four HPE algorithms (Andriluka et al. [5], Eichner and Ferrari [26], Sapp et al. [80], Yang and Ramanan [103]) at three different CPC thresholds. The evaluation criteria is the area under the ROC curve (AUC). BL is the AUC of the baseline and PA is the AUC of our pose evaluator.

<table>
<thead>
<tr>
<th>HPE</th>
<th>CPC 0.2</th>
<th></th>
<th>CPC 0.3</th>
<th></th>
<th>CPC 0.4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BL</td>
<td>PA</td>
<td>BL</td>
<td>PA</td>
<td>BL</td>
<td>PA</td>
</tr>
<tr>
<td>Andriluka [5]</td>
<td>56.7</td>
<td>90.0</td>
<td>56.2</td>
<td>90.0</td>
<td>55.8</td>
<td>89.2</td>
</tr>
<tr>
<td>Eichner [26]</td>
<td>84.3</td>
<td>92.6</td>
<td>81.6</td>
<td>91.6</td>
<td>80.5</td>
<td>90.9</td>
</tr>
<tr>
<td>Sapp [80]</td>
<td>76.5</td>
<td>82.5</td>
<td>76.5</td>
<td>83.0</td>
<td>76.9</td>
<td>83.5</td>
</tr>
<tr>
<td>Yang [103]</td>
<td>79.5</td>
<td>83.7</td>
<td>78.4</td>
<td>81.5</td>
<td>78.4</td>
<td>81.2</td>
</tr>
</tbody>
</table>

Figure 4.8: (a) Performance of HPE evaluator in regime A: (no false positives used in training or testing). The ROC curve shows that the evaluator can successfully predict whether an estimated pose has CPC < 0.3. (b) Performance of HPE evaluator in Regime B: (false positives included in training and testing).

Figures 4.8a and 4.8b show performance for regime A and regime B respectively. The ROC curves are plotted using the score of the evaluator, and the summary measure is the Area Under the Curve (AUC). The evaluator is compared to a relevant baseline that uses the HPE score as a confidence measure (i.e. the energy of the most probable (MAP) configuration $L^*$). For Andriluka et al. [5], the baseline is the sum over all parts of the maximum value of the marginal distribution for a part. The plots demonstrate that the evaluator works well, and outperforms the baseline for all the HPE methods. Since all the HPE algorithms use the same upper body detections, their performance can be compared fairly.

To test the sensitivity to the 0.3 CPC threshold, we learn a pose evaluator also using CPC thresholds 0.2 and 0.4 under the regime B. Table 4.4 shows the performance of the pose evaluator for different CPC thresholds over all the HPE algorithms. Again, our pose evaluator shows significant improvements over the baseline in all cases. The improvement of AUC for Andriluka et al. [5] is over 33.5. We
Figure 4.9: Example evaluations. The pose estimates in the first two rows are correctly classified as successes by our pose evaluator. The last two rows are correctly classified as failures. The pose evaluator is learnt using the regime B and with a CPC threshold of 0.3. Poses in rows 1,3 are estimated by Eichner and Ferrari [26], and poses in rows 2,4 are estimated by Yang and Ramanan [103].

believe that this massive increase is due to the suboptimal inference method used for computing the best configuration. For Eichner and Ferrari [26], Sapp [80], and Yang and Ramanan [103] our pose evaluator brings an average increase of 9.6, 6.4 and 3.4 respectively across the CPC thresholds. Interestingly, our pose evaluator has a similar performance across different CPC thresholds.

Our pose evaluator successfully detects cases where the HPE algorithms succeed or fail, as shown in figure 4.9. In general, an HPE algorithm fails where there is self-occlusion or occlusion from other objects, when the person is very close to the image borders or at an extreme scale, or when they appear in very cluttered surroundings. Analyzing the learnt weight vector from the linear SVM, we see that the HPE evaluator too has identified these as the failure cases (sec. 4.2.3). The magnitude of the components of the weight vector suggests that the features based on marginal probability and the location of the detection window are very important in distinguishing the positive samples from the negatives. On the other hand, the weights of the upper body detector, image intensity and image gradient are low and thus do not have much impact.
4.4 Applying Pose Evaluator

In many applications, we are interested only in fully correct pose estimates, i.e. with all parts correctly estimated. For instance, a pose estimate with some incorrect parts can severely lower the perceived performance of a retrieval system, if it appears high up in the ranked list. Here we use the assessment criterion of [26] and define a part as correctly estimated if its segment end-points lie within 50% of the length of the ground-truth annotation.

In the following experiments we only consider two HPE algorithms [26, 103] as their computational speed is high, but it can easily be extended to other algorithms [5, 80] as well. Unfortunately, for both HPE algorithms the percentage of fully correct pose estimates is very low (10.9% and 17.6% from table 4.3). Note that we are applying a very severe test for a correct pose – that all parts are correct. In contrast, the quantitative measure used to assess performance in [26, 103] counts the average number of correctly estimated parts (PCP). For example, a pose with 5 out of 6 parts correctly estimated has a PCP score of $5/6 = 0.83$, but scores zero under our criterion.

In order to improve the percentage of correct pose estimates in a given set of estimates, we employ the pose evaluator in three different ways. Before applying these methods, all the images are first processed with the upper body detector [32]. Then, on each upper body detection the two pose estimation algorithms Eichner and Ferrari [26], Yang and Ramanan [103] are run. The confidence scores given by these algorithms are normalized to $[0, 1]$ range using Platt [73]. The parameters for the normalization are learned separately for each HPE algorithm.

4.4.1 Methods

For different problems, the number of algorithms with good performance may vary. To address this variation the following three methods have been developed which work with one or multiple algorithms. ‘Filter I’ works with one algorithm, while ‘Filter II’ and ‘Hybrid pose estimation’ methods work with two or more algorithms.

Filter I: Removing samples with low score. In this method, we use the pose evaluator to remove incorrect pose estimates. By construction, the pose evaluator tends to give a high confidence score to correctly estimated poses and a low score to incorrect ones. Hence, we use the pose evaluator to give a confidence score to each pose estimate in the input set. We can then remove pose estimates by thresholding this confidence score. Below we use the confidence score in other ways as well.

Filter II: Agreement of pose estimates. In this method, we consider the agreement between poses produced by two different HPE algorithms on the same human detection. The intuition is, while the incorrect pose estimates of the algorithms can be very different, the correct pose estimates would be the same. Since the algorithms are based on different features and inference algorithms, they are rather complementary in their failure modes. Hence we expect that our agreement criterion should reliably identify correct pose estimates.
We consider two poses to be in agreement if the PCP between them is perfect (i.e. 1.0). Suppose the algorithms of Eichner and Ferrari [26] and Yang and Ramanan [103] generate pose estimates A1 and A2 respectively, then A1 and A2 are in agreement if the PCP between them is perfect (PCP=1.0). If this is the case, then the pair (A1, A2) is added to the agreement set. The new confidence score of the pair (A1, A2) is the linear combination of both the confidence scores given by the pose evaluator.

**Hybrid pose estimation:** In this method, we make another use of outputs from multiple HPE algorithms on each detected person. The key observation is that often different algorithms fail on different images. Hence, chances are higher that at least one algorithm out of several has correctly estimated the pose, than when relying on a single algorithm. We use our pose evaluator to identify the correct output from among these multiple pose estimates. This is done by taking the highest scoring pose estimates from all the pose estimates belonging to a person.

Note that after processing the input set using the above three methods, we can either threshold the pose estimates or rank them using the new confidence scores assigned by the respective methods.

### 4.4.2 Results

We evaluate the three methods proposed in the previous subsection individually and then compare their relative performances. The three methods are evaluated on the same test set (section 4.3.1) used to test the pose evaluator.

The evaluation measure used is average precision, which is the area under the precision recall plot. First the pose estimates, denoted by S, are sorted based on the confidence score assigned by a method described in previous subsection. At each possible recall point, precision is computed and precision-recall curve is plotted. Precision is defined as the total number of positives in S divided by the cardinality of S. Recall is defined as the total number of positives in S divided by the total number of ground truth stickmen.

**Filter I:** This method is evaluated by measuring average precision over the test set, on a range of thresholds, by removing the pose estimates with pose evaluator scores less than threshold. Figure 4.10a shows that the average precision improved significantly from, 16.6% (before filtering) to 39.2% (after filtering) for [26] and from 26.4% (before filtering) to 34.0% (after filtering) for [103].

**Filter II:** This method is evaluated by first performing the procedure described in section 4.4.1 to obtain an agreement set S. An element e = (A1, A2) in S is deemed to be positive if both A1 and A2, pose estimates from [26, 103] respectively, are in full agreement (PCP=1.0) with the ground-truth stickman. The confidence score, as mentioned in the previous section, is the linear combination of baseline scores \( \text{baseline}_{EF}, \text{baseline}_{YR} \) of both the HPE algorithms [26, 103] respectively. Empirically we have found that the linear combination of \( 0 \ast \text{baseline}_{EF} + 1 \ast \text{baseline}_{YR} \) gives the best results.

Figure 4.10b shows that the agreement operation, with legend ‘Intersection@PCP=1.00’, improves precision significantly. It outperforms both baseline [26, 103] at most recall points. Since this method rejects a considerable number of pose estimates, the curves end before reaching recall 1.0. This method is expected to have high precision and low recall as it aggressively filters out incorrect pose estimates and
in the process loses some correct pose estimates too. We also tried weaker criteria requiring agreement in at least five parts (PCP ≥ 0.83) or at least four parts (PCP ≥ 0.67). As figure 4.10b suggests, these weaker criteria improve the best attainable recall while worsening precision significantly.

**Hybrid pose estimation:** Finally, this method is evaluated by combining the pose estimates of [26] and [103] as described in section 4.4.1. Using the new confidence score assigned by this method, average precision is computed. Plot 4.10c shows that the average precision improved greatly from 16.6% [26] and 26.4% [103] to 53.6%.

Comparing the three methods we can see from figure 4.10 that the ‘Filter II’ method has high precision and low recall, ‘Filter I’ method has good precision and tolerable recall and ‘Hybrid pose estimation’ method has good precision and high recall (relative to the HPE algorithms). Hence, when only one HPE algorithm is available, the ‘Filter I’ method is a better option than using the HPE output ‘as is’. When multiple HPE algorithms are available, the ‘Hybrid pose estimation’ algorithm is the best option, as it outperforms the ‘Filter II’ method.

### 4.5 Pose Representation

From the human pose estimation algorithm, we obtain pose estimates consisting of line segments corresponding to the upper body parts namely head, torso, right and left arms. These depend on the size of the person and their location. To compare two pose estimates we require a representation that is invariant to scale and location in the image. Previous representations used for retrieval purposes were high dimensional, for example the three descriptors proposed by [40] have 15360, 1449 and 1920 dimensions. These descriptors are very high dimensional because they represent either full probability distributions over possible part positions and orientations, or soft-segmentations of the parts.
Figure 4.11: **Pose Representation:** The two poses A and B differ in two parts, the upper and lower left arms. The pose representation based on the angles clearly distinguishes pose A from pose B. The endpoints of each stickmen are consistently numbered.

In contrast, we use a simple and effective representation based on a single absolute orientation of each part. The orientation of parts are independent of the location and the size of the human and are not affected by variations in the relative length and positions of parts.

When comparing two poses, we would like to form a distance measure based on the sum of differences of angles of corresponding parts. To achieve this we encode the angle $\theta$ of each part as the 2D vector $(\cos \theta, \sin \theta)$. For six parts this results in a 12 dimensional feature vector. Then the Euclidean distance between two such representations gives the cosine of the angular difference. This is elaborated in more detail below. Figure 4.11 illustrates the pose representation.

### 4.6 Summary

In this chapter we described a pose representation based on the output of human pose estimation algorithms. We highlighted the fact that human pose estimation algorithms are very unreliable. To improve the reliability, we have introduced the concept of an evaluator algorithm which we developed for human pose estimation methods. Our evaluator accurately predicts if the vision algorithm has succeeded or not when usually for such algorithms no confidence score (only a MAP score) is provided. We have also shown that the evaluator can be used to effectively filter incorrect pose estimates, fuse outputs from different pose estimation algorithms and improve the quality of pose representations derived from pose estimation algorithms.

More generally, we have cast self-evaluation as a binary classification problem, using a threshold on the quality evaluator output to determine successes and failures of the HPE algorithm. An alternative
approach would be to learn an evaluator by regressing to the quality measure (CPC) of the pose estimate. We could also improve the learning framework using a non-linear SVM.

We believe that our evaluator has wide applicability. It works for any part-based model with minimal adaptation, no matter what the parts and their state space are. We have shown this in the work, by applying our method to various pose estimators [5, 26, 80, 103] with different parts and state spaces. A similar methodology to the one given here could be used to engineer evaluator algorithms for other human pose estimation methods e.g. using poselets [13], and also for other visual tasks such as object detection (where success can be measured by an overlap score).

In the next chapter, we describe a prototypical large scale human pose retrieval system using the pose representation developed in this chapter. All the aspects pertaining to a retrieval system such as query modality, user interface and nearest neighbor search in high dimensional space are explored in detail.