Chapter 4

Articulated Pose Estimation

In this chapter we describe a new approach to estimation of articulated pose of humans from single monocular images. Our approach is example-based: it reduces the problem of recovering the pose to a database search under L_1 in the embedding space, which is carried out extremely fast using LSH. The embedding is constructed based on edge direction histograms, using the algorithms presented in Chapter 3. Underlying this construction is the definition of a similarity concept under which two images of people are similar if the underlying poses are, and learning an embedding that is sensitive to that similarity.

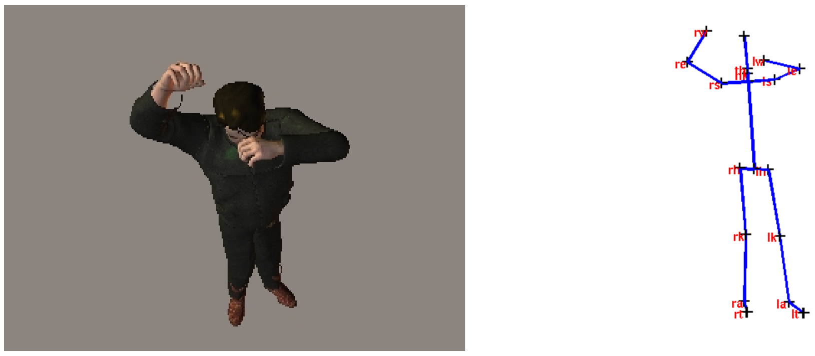
We start with describing the problem domain and presenting our approach to it in a nutshell in Section 4.1, and cover some related work in Section 4.2. Section 4.3 gives the details of the representation and the learning problems defined for the task. Experimental results in two estimation tasks are described in Sections 4.4 and 4.5. In Chapter 5 we discuss the integration of our approach to single-frame pose estimation into a tracking framework.

4.1 The problem domain

The articulated pose estimation problem is formulated as follows. We are given an image which contains a human body.[1] We also have an *articulation model*–a model of the body that describes the current 3D body configuration in terms of a set of *limbs* and rotational *joints* that connect them into a tree structure.

This model is illustrated in Figure 4-1. The image on the left is not a photograph of a real person but a synthetically generated image of a humanoid model obtained with a computer graphics program Poser [29]. This image corresponds to the articulated model in the left part of the figure. The model is shown by plotting 2D projections of 20 key joints (crosses) and the lines connecting them, that roughly correspond to limbs. This model may be described by 60 numbers, namely the (X, Y, Z) coordinates of the joints (an alternative form of describing the model would be in terms of articulated angles, which we will discuss later.) In fact, there are hundreds of parameters in

[1]The presented framework can be applied to any articulated body, but estimating pose of humans is by far the most important task of this sort.



(a) Image of a body (b) Corresponding articulated model

Figure 4-1: A (synthetic) image of a person and the corresponding articulated model. The goal of pose estimation is to derive the representation from the image on the left. Crosses show key joints, labeled with abbreviations. **l**/**r**: left/right, **t**: big toe, **a**: ankle, **k**: knee, **h**: hipbone, **s**: shoulder, **e**: elbow, **w**: wrist. Additional parts are the base of the neck **nk**, the base of the skull **th** and the top of the skull (not labeled).

addition to these 60 numbers that affect the resulting image: the articulated pose of additional body parts not accounted for by this coarse model, such as fingers; shape of the actual body parts (the model, so to speak, describes the "bones", but not the flesh); facial expression; hair style; clothing; illumination etc. Added to that could be the parameters that describe the scene, the objects in the background etc. The goal of a computer graphics program like POSER is to start with these parameters and produce a realistic image, that is, to go from the right half of Figure 4-1 to the left half. The goal of computer vision is the opposite. In the context of articulated pose estimation this goal is to start from the left half (the image), and recover the relevant parameters (the right half) of the representation that "generated" the image, while ignoring the *nuisance parameters*–all those additional aspects of the visual scene listed above. When the image is actually synthetically generated the success of this task is easy to measure, since we have access to the ground truth. For real images such evaluation is more difficult. When measurements of the underlying pose are available, for example obtained using a motion capture device at the same time as the images are taken, this may be done in a precise fashion.[2] In other cases this may be subjective, or it may depend on the success of a "downstream" application that relies on the estimated pose (we discuss some applications in the next section and in Chapter 5.)

[2]However, special caution is required to make sure the motion capture setup, e.g. special clothing or visible sensors, is not used by the estimation algorithm to "cheat".

4.2 Background on pose estimation

There exists a large body of literature on the estimation of the pose of articulated bodies. We only focus here on work most related to our approach. It should also be noted that much more attention has been given to the task of *articulated tracking* of humans: recovering the sequences of articulated poses from a video showing a moving person. This task is usually approached in a qualitatively different way from single-frame pose estimation. In particular, tracking algorithms (with almost no exceptions) rely on the assumption of manual initialization. While the tracking setup is in some ways more challenging than the single-frame one, it also allows access to provides valuable cues from motion that are not available in a static task. This may allow, in particular, to disambiguate certain situations which are very difficult or even impossible to disambiguate with a single frame. We will not discuss tracking here, but in Chapter 5 we will describe tracking algorithms that integrate our approach to single-frame pose estimation with a tracking setup, allowing us to relax or even abandon the initialization assumption.

Providing automatic initialization (and re-initialization throughout the sequence) for tracking is among the most important applications of single frame pose estimation. In fact, having a perfect pose estimator would eliminate the need for specialized tracking algorithms, since the accurate pose recovery would simply be done in every frame. Of course, this is not possible since single-frame estimation is ill-posed: in many "interesting" activities there is a great deal of occlusion of some body parts by others, there is often ambiguity related to symmetry, mirror reflections etc. Nevertheless the ability to recover pose from a single image is crucial for successful tracking. We discuss this in more detail in Chapter 5.

Much of the work has relied on deterministic methods guided on the known geometry of the articulated body. In [111] 3D pose is recovered from the 2D projections of a number of known feature points on an articulated body. Other efficient algorithms for matching articulated patterns are given in [45, 94, 88]. All of these approaches assume that detectors are available for specific feature locations, and that a global model of the articulation is available. Another family of approaches can somewhat relax these assumptions, at the cost of relying on the availability of multiple views [58].

Other techniques are based on statistical learning approaches. In [87] pose estimation is reduced to contour shape matching using *shape context* features. In [95], the mapping of a silhouette to 3D pose is learned using multi-view training data. These techniques were successful, but they were restricted to contour features and generally unable to use appearance within a silhouette. Some methods explicitly work with silhouettes only [40, 2] but those, due to a rather impoverished representation that greatly increases ambiguity, are usually restricted to a specific type of activity (walking is particularly popular.)

In [6] a hand image is matched to a large database of rendered forms, using a sophisticated similarity measure on image features. This work is most similar to ours and in part inspired our approach to pose estimation. However, the complexity of nearest neighbor search makes this approach difficult to apply to the very large numbers of examples needed for general articulated pose estimation with image-based distance metrics.

Finally, we should emphasize that the task of pose estimation we are considering is decoupled from the tasks of *detection* and *localization*, i.e., determining whether an image contains a person and finding the specific portion of the image occupied by the person. There are a number of methods for carrying out those tasks, and we will assume that localization is solved by an external algorithm. Specific arrangements for obtaining this information in our experiments is described in Sections 4.4 and Section 4.5.

4.3 Example-based pose estimation

We approach pose estimation as a regression task, and develop an example-based approach to solving it. As described in Section 2.2.1, we can define a similarity concept $\mathcal{S}_p$ corresponding to pose similarity. We assume that we have access to a large and representative[3] database of images labeled with the corresponding poses. Then, the pose in a query image $\mathbf{x}_0$ can be estimated in by the following two steps:

- Find in the database some examples of poses similar to the unknown pose in $\mathbf{x}_0$.

- Using the retrieved examples, infer the pose in $\mathbf{x}_0$.

This fairly vague recipe is detailed in the sections below.

4.3.1 Pose-sensitive similarity

Suppose that a pose is represented by a parameter vector θ (we discuss some parameterizations below). Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two images depicting people whose articulated poses are, respectively, θ_1 and θ_2. Then, we define

$$\mathcal{S}_{p,R}(\mathbf{x}_1, \mathbf{x}_2) = +1 \;\Leftrightarrow\; \mathcal{D}_\theta(\theta_1, \theta_2) \leq R. \tag{4.1}$$

This is a generic similarity "template", and the precise definition depends on two parameters: the distance $\mathcal{D}_\theta$ used to compare poses, and the appropriate threshold R on that distance. The threshold could be set in two ways. The first is by finding R which meets some perceptual criteria: if $\mathcal{D}_\theta(\theta_1, \theta_2) \leq R$, then human observers will generally agree that the two poses "look similar", or are similar for the purpose of a particular application. Our approach to learning similarity from examples, developed in Chapter 3, is perfectly suited for such a definition since all it requires is a set of examples of similar pairs–which in this case may be supplied by human observers. A second method of setting R is by means of validation tuning with a specific estimation

[3]In the sense that for a random pose drawn from the distribution of poses, there is, with high probability, an example with a similar pose, under the relevant definition of similarity discussed in this section.

algorithm. That is, if the goal is to recover pose as precisely as possible,[4] and the estimation algorithm relies on similarity defined in (4.1), then we may look for R that minimizes the final error.

As for $\mathcal{D}_\theta$, there are two avenues for defining it, and the choice depends on the representation of the articulated model. A common representation, common in computer graphics and animation, is by *joint angles*[93]. Consider a directed graph representation of an articulated tree, where each node corresponds to a joint (we use the term joint loosely to refer to any rigid point in the model, so that, for instance, the top of the skull is also considered a "joint".) Edges leaving the node correspond to the limbs connected to that joint, and they connect it to the joints on the other side of the limb. Then the entire configuration of the model in 3D is given by a set of 3D rotation parameters in each joint plus the global position and orientation of the root, which is usually at the hip joint. This representation is convenient to describe articulation, and especially to parametrize articulated motion. Also, it describes the body articulation independently of the sizes of actual limbs. However it makes defining distances quite cumbersome. For instance, a 20 degree change in an angle may affect the global position of body parts very little if it is in a finger, or very much if it is in the hip.

For this representation, we use the *mean cosine deviation* distance $\mathcal{D}_{cos}$:

$$\mathcal{D}_{cos}(\theta_1, \theta_2) = \sum_{i=1}^{m} \left(1 - \cos(\theta_1^i - \theta_2^i)\right) \tag{4.2}$$

The second representation is in terms of 3D joint locations [57]. If there are L joints in the model, then the pose θ_i is fully described by $\theta_i = [\theta_i^1, \ldots, \theta_i^L]$, where the location of the j-th joint is given by $\theta^j = [\theta_{x,i}^j, \theta_{y,i}^j, \theta_{z,i}^j]^T \in \mathbb{R}^3$. This representation is somewhat redundant, since there are strong constraints on the relative locations of neighboring limbs, however it is very explicit and thus convenient for manipulating and comparing poses.

For this representation, we define the maximum deviation distance $\mathcal{D}_D$ by the maximum L_1 distance between any two corresponding joints in 3D:

$$\mathcal{D}_D(\theta_1, \theta_2) = \max_{1 \le j \le L} \sum_{d \in x,y,z} |\theta_{d,1}^j - \theta_{d,2}^j|. \tag{4.3}$$

In accordance with the approach we have outlined above, we will learn an embedding of the images space into a new space H, such that for two images $\mathbf{x}_1, \mathbf{x}_2$ and the corresponding poses θ_1, θ_2, $\|H(\mathbf{x}_1) - H(\mathbf{x}_2)\|$ is, with high probability, low if $\mathcal{D}_D(\theta_1, \theta_2) \le R$.

[4]Note that this is rarely the real goal of an application; for instance, in an activity recognition scenario, or for understanding gestures, an error of a few degrees or a few centimeters relative to the "ground truth" is rarely a problem.

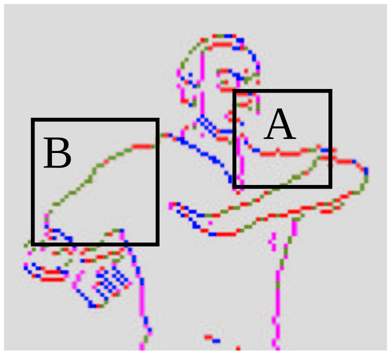


Figure 4-2: Illustration of the edge direction histogram (EDH) representation. Colors correspond to detected edge orientation red=0, green=$\pi/4$, purple=$\pi/2$ and blue=$3\pi/4$.

4.3.2 Image representation

Before we approach the learning task, we need to design the representation of the input space $\mathcal{X}$. The simplest decision would be to simply use the pixels of the image. However it is clearly not very helpful, due to a large effect of the nuisance parameters (color and illumination in particular) on the pixel intensities, and we would benefit from a representation that is more invariant to nuisance parameters while capturing information useful for inferring pose. In this chapter we will use the representation by multi-scale *edge direction histograms* (EDH) [68], often used in image analysis and retrieval, but until now it has not, to our knowledge, been used for pose analysis.

In order to compute EDH, we apply an edge detector of choice (we have used the Sobel detector [54]) to obtain an edge map, i.e. a binary image in which the value of a pixel is 1 if a detected edge passes through it. Next, each detected edge pixel is classified into one or more of four direction bins: $\pi/8, 3\pi/8, 5\pi/8, 7\pi/8$. This is done by applying a local gradient operator at each of the four orientations, and thresholding the response. Then, the histograms of direction bins are computed within sliding square windows of varying sizes (scales) placed at multiple locations in the image; the scales and the location grid are parameters to be set. This yields four integer values (the counts for the four direction bins) for each scale and location. The resulting multi-scale EDH is obtained by concatenating these values in a fixed order. Figure 4-2 illustrates the EDH representation; each of subwindows A and B contributes four numbers, calculated by counting edge pixels of four colors within the subwindow.[5]

Assuming, as we do, that the person localization task is solved for us and the image is centered on the bounding box of the body, a reasonable measure of similarity to apply to this representation is the L_1 distance, since a particular bin in the histogram corresponds to a roughly fixed location on the body. It is interesting to note the connection of this distance to the Hausdorff and Chamfer distances often used to

[5]Some pixels, in particular the ones at edge intersections, may have multiple colors, i.e. multiple orientations, assigned to them.

compare silhouettes or edge images [11]. A related distance is the Earth-Mover's distance[55].

Another interesting connection is to shape contexts [11], that have been used for pose estimation among other tasks [86, 87]

4.3.3 Obtaining labeled data

Our approach relies on the availability of a large database of images labeled with poses. Such a database may be constructed either by means of computer graphics package, such as Poser. or by recording data from human subjects. The synthetic generation is an appealing option since it is extremely cheap, can provide an arbitrarily large number of examples, and makes it easy to include as much variability in the data as desired (subject to model limitations of the software.) Importantly, it also provides accurate ground truth of the pose for every image. The resulting images can be quite realistic in terms of pose appearance (see Figures 4-3 and 4-6 for some examples).

Alternatively, such a database could also be created by recording images of real people in a variety of poses, along with the poses themselves measured by one of the available methods for that (usually based on instrumenting the actor with some sort of sensors.) However, this may be extremely expensive, labor-intensive and time-consuming. This may be possible for a constrained set of poses, for instance associated with a particular task or activity. If the goal is to have a very large database highly representative of the general pose space, this approach is probably infeasible, and even more so if we also want to include a significant variation in nuisance parameters in the data. One potential advantage of such a database, of course, is that the real training images may, in some sense, look more "like" the real test images the system would encounter. However in our opinion the state-of-the-art in computer graphics, as exemplified by Poser, removes this concern since the synthetic images are close in quality to the real ones, at least for the single-frame pose estimation purposes.[6] A more important advantage of a human-based database is in the realistic nature of the poses it contains, both in terms of the distribution and in terms of attainable configurations.

Fortunately, there is a way to have the best of both worlds. A set of poses can be recorded with a motion capture setup, and then used to create a large set of synthetic images by changing the viewpoint, slightly perturbing the poses, and randomly assigning the nuisance parameters. This is the approach taken to obtain the training data used in experiments described in Section 4.5 and in Chapter 5.

4.4 Estimating upper body pose

The experiments described in this section[7] deal with estimating only a partial pose, namely that of the upper body. The joints model specifies the location of shoulders, elbows and wrists. It is assumed that the person in the image is visible from about

[6]This may not yet be the case for synthetic rendering of motion!

[7]This section is based on the work published in [105]

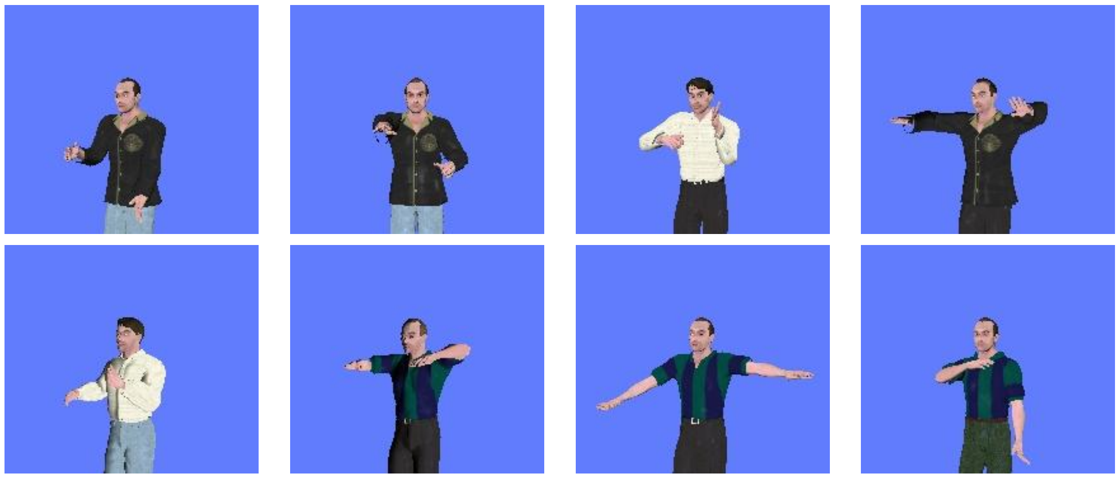


Figure 4-3: Example training images for upper body pose estimation

the knee level up and is standing in an upright posture. The orientation (yaw) of the body is not constrained, and may vary between the two profile views, $\pm 90^o$.

4.4.1 Training data

The database of poses contains 500,000 images obtained by sampling uniformly at random the space of articulation angles, applying a feasibility correction algorithm of Poser (to prevent configurations which are either anatomically impossible or physically impossible, e.g. surface intersections), and rendering a 180×200 pixel image with randomly assigned nuisance parameters: illumination (obtained by modeling 4 random light sources), hair style, clothing, and hand configuration. As stated above, we assume that the body has been segmented from background, scaled, and centered in the image. Thus no background detail was generated, so the figures are on a uniform background. Figure 4-3 shows some examples.

4.4.2 The learning setup

The EDH representation was constructed with windows of sizes 8, 16 and 32, with each window sliding through locations spaced by half its size, yielding 11,728 histogram bins per image. With two bytes to represent each histogram bin, this requires above 11 Gigabytes to record the EDH for the full database.

Pose similarity was defined by setting a threshold of 0.5 on the $\mathcal{D}_{cos}$ between poses. This value was chosen by inspection, as it corresponded to a good cutoff between perceptually similar and dissimilar pairs of poses. Not surprisingly, similarity in this domain is a rare event; the similarity rate ρ defined in Section 3.2.4, measured on a million random pairs constructed over the training data, was only 0.0005.

Using the EDH representation as the input space $\mathcal{X}$, we constructed a training set for SSC: 100,000 positive examples and 1,000,000 negative examples. The larger number of the negative examples was motivated by the unbalanced nature of the

Model	$k = 7$	$k = 12$	$k = 50$
k-NN	0.882 (0.39)	0.844 (0.36)	0.814 (0.31)
Linear	0.957 (0.47)	0.968 (0.49)	1.284 (0.69)
const LWR	0.882 (0.39)	0.843 (0.36)	0.810 (0.31)
linear LWR	0.885 (0.40)	0.843 (0.36)	0.808 (0.31)
robust const LWR	0.930 (0.49)	0.825 (0.41)	0.755 (0.32)
robust linear LWR	1.029 (0.56)	0.883 (0.46)	0.738 (0.33)

Table 4.1: Mean estimation error for 1000 synthetic test images, in terms of $\mathcal{D}_{cos}$. Standard deviation shown in parentheses. Not shown are the baseline error of 1-NN, 1.614 (0.88), and of the exact 1-NN based on L_1 in $\mathcal{X}$, 1.659. LWR stands for locally-weighted regression, see Section 2.2.

problem, discussed in Chapter 3.

We evaluated a number of TP-FP gap values on a small validation set, and set the lower bound on the gap g to 0.25. With that gap bound, SSC selected 213 dimensions. Thus, the size of the database could be reduced, with the most economical data storage, from 11 Gigabytes to less than 14 Megabytes (recall that the dimensions produced by SSC are bit valued.) This data structure was then indexed by LSH, with l=80 tables and $k = 19$ bits per hash key. Note that the application of algorithm 3 (p. 2.4.2) is particularly simple on the bit-valued embedding H since each dimension only has one possible threshold. Thus the application of SSC with subsequent indexing by LSH may be seen as simply learning of an appropriate family of LSH functions.

We also tested the semi-supervised version of SSC described in Chapter 3. As expected for the low similarity rate in this case, the results were very similar to the results with the fully supervised version: we obtained 221 dimensions, with 97% overlap with the dimensions learned with the supervised algorithm. Thus we get essentially identical results with more than 10 times reduction in learning time (since the semi-supervised algorithm uses only 1/11 of the training examples used in the fully-supervised one.)

4.4.3 Results

To quantitatively evaluate the algorithm's performance, we tested it on 1000 synthetic images, generated from the same model, so that the ground truth is available. Table 4.1 summarized the results with different methods of fitting a local model; 'linear' refers to a non-weighted linear model fit to the neighborhood. The average size of the candidate set C found by LSH (i.e. the union of the buckets in the hash tables) was 5300 examples, about 1% of the data. We found that in almost all cases, the true nearest neighbors under $\mathcal{D}_H$ were among the candidates, which means that we do not pay significant cost for the speedup obtained with LSH.

The locally-weighted regression (LWR) [7] model was tested with zeroth-order, or constant, model (i.e., weighted average of the neighbors) and first-order, or linear,

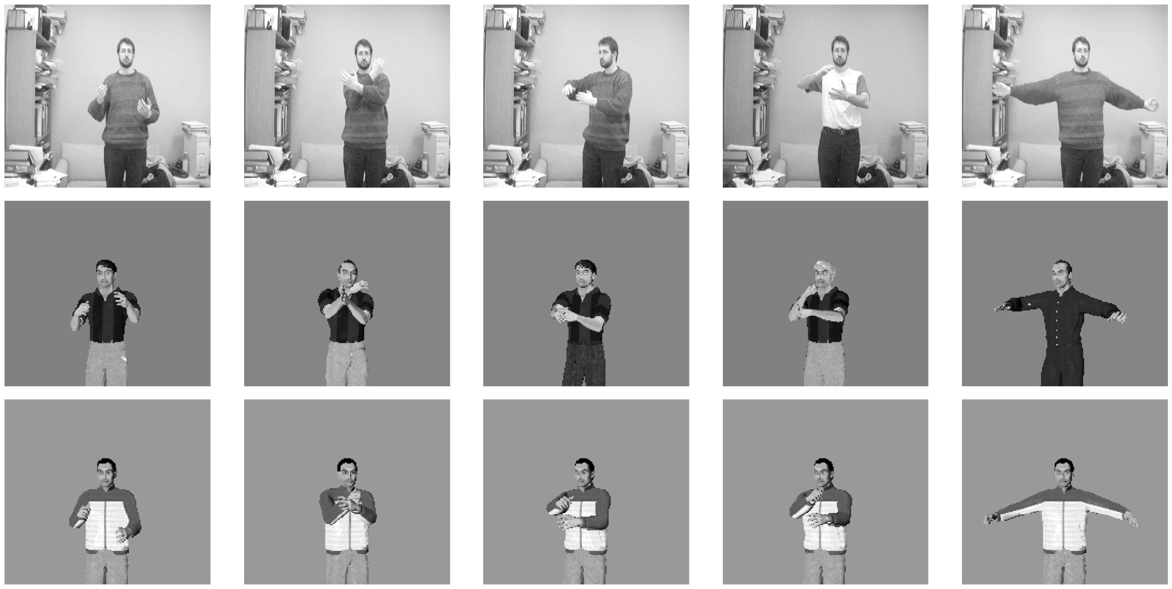


Figure 4-4: Examples of upper body pose estimation (Section 4.4). Top row: input images. Middle row: top matches with LSH on the SSC embedding. Bottom row: robust constant LWR estimate based on 12 NN. Note that the images in the bottom row are not in the training database - these are rendered only to illustrate the pose estimate obtained by LWR.

model (i.e., weighted linear fit.) The robust LWR [22] re-weighted the neighbors in 5 iterations. The purpose of robust LWR, as explained in Section 2.2, is to reduce the influence of the outliers (examples with high residual under the current model fit) by iteratively decreasing their weights.

The results confirm some intuitive expectations. As the number of approximate neighbors used to construct the local model increases, the non-weighted model suffers from outliers, while the LWR model improves; the gain is especially high for the robust LWR. Since higher-order models require more examples for a good fit, the order-1 LWR only becomes better for large neighborhood sizes. Overall, these results show consistent advantage to LWR. Note that the robust linear LWR with 50 NN is on average more than twice better than the baseline 1-NN estimator.

We also tested the algorithm on 800 images of a real person; images were processed by a simple segmentation and alignment program, using a statistical color model of the static background and thresholding by intensity change. Figure 4-4 shows a few examples of pose estimation on real images. Note that the results in the bottom row are not images from the database, but a visualization of the pose estimated with robust linear LWR on 12-NN found by LSH; we used a Gaussian kernel with the bandwidth set to the d_X distance to the 12-th neighbor. In some cases (e.g. leftmost column in Figure 4-5), there is a dramatic improvement versus the estimate based on the single NN. The number of candidates examined by LSH was significantly lower than for the synthetic images - about 2000, or less than .5% of the database. This is expected since the real images differ from the synthetic ones in many subtle ways.

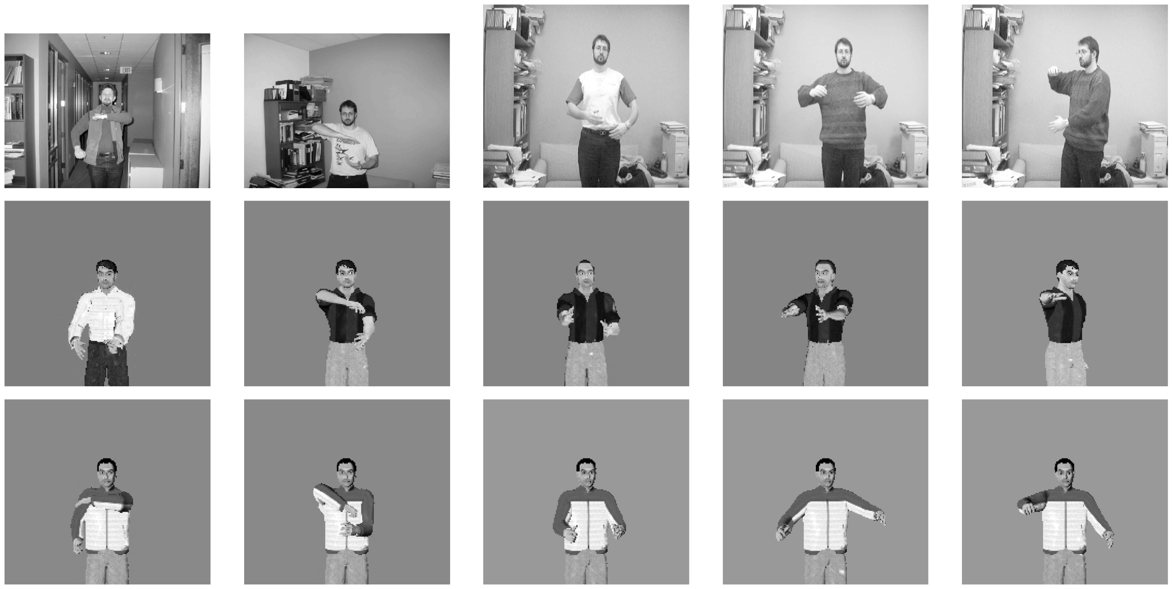


Figure 4-5: More examples, including typical "errors"; see legend of Figure 4-4. Note the gross error in the leftmost column, corrected by LWR. Examples in the right two columns are among the ones with most severe error in the test set.

It takes an unoptimized Matlab program less than 2 seconds to produce the pose estimate. This is a dramatic improvement over searching the entire database for the exact NN under L_1 in the embedding space, which takes more than 5 minutes per query, and in most cases produces the same top matches as the LSH. Note that exact search under L_1 distance in $\mathcal{X}$ (EDH) would take a number of days, in particular due to the enormous size of the database mentioned above.

Lacking ground truth for these images, we relied on visual inspection of the pose for evaluation. For about 2/3 of the examples the pose estimate was judged accurate; Figures 4-4 and 4-5 show a number of examples of typical estimates. On the remaining examples it was deemed inaccurate, on some examples the error was quite significant. Figures 4-4 and 4-5 show a number of examples, including two definite failures. Note that in some cases the approximate nearest neighbor is a poor pose estimate, while robust LWR yields a much better fit.

Nevertheless this system clearly can be improved. We can identify three sources of failure. One, not directly related to the learning and estimation procedures, is imperfect segmentation and alignment. The other potential reason is the suboptimal set of dimensions found by SSC (perhaps due to a poor choice of the gap bound); we suspect that 213 dimensions in the embedding is not rich enough a representation. The third problem is related to the limitations of the synthetic training set, in terms of coverage and representativeness of the problem domain. The experiment reported in the next section addressed some of these issues.

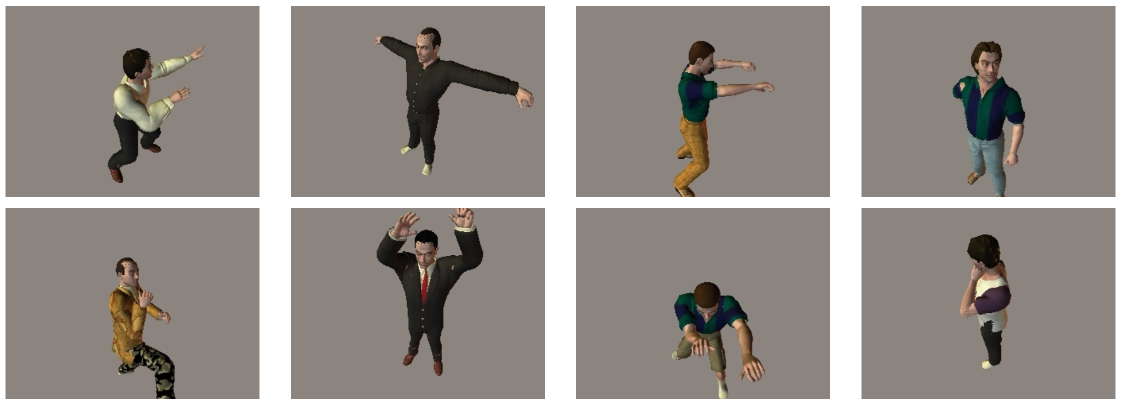


Figure 4-6: Examples of images in the motion capture-based repository of full body pose used in the experiments in Section 4.5.

4.5 Estimating full body pose

In this experiment we estimate full body pose, with the articulated model containing 60 parameters (this is the model illustrated in Figure 4-1(b).)

4.5.1 Training data

To improve the quality of the database we used the motion capture sequence freely available from [41]. The database contains over 600 sequences recorded from a variety of activities from everyday life (walking, greeting, brushing teeth), athletics (soccer, martial arts), etc. We collected 550,000 unique poses (with $\mathcal{D}_D$ between any two poses, as defined in (4.3), at least 1cm) and rendered a 240×320 pixel image from each pose at three random yaws, yielding a repository of 1,650,000 images labeled with the ground truth pose. The figure in each image is rendered at a random 2D location within the virtual scene, with up to 1m translation, in order to represent variability and with the intent to make the resulting estimator invariant to moderate translations (the 2D location is considered a nuisance parameter.) Figure 4-6 shows some examples of the images in this repository.

From each image we extracted the bounding box of the silhouette (using the fact that these synthetically generated images have known segmentation and thus the silhouette mask is available), and computed the EDH representation as described above, yielding 13,076 bins in a histogram.

4.5.2 Learning setup and results

We selected 60,000 images from the repository, constrained to upright postures. From these, we formed 20,000 positive pairs, subject to the similarity defined as in (4.1) with $\mathcal{D}_D$ as the pose distance and $r = 3cm$.

We then applied a semi-supervised version of BoostPro, using linear projections over two dimensions. That is, each dimension of the embedding is obtained by taking

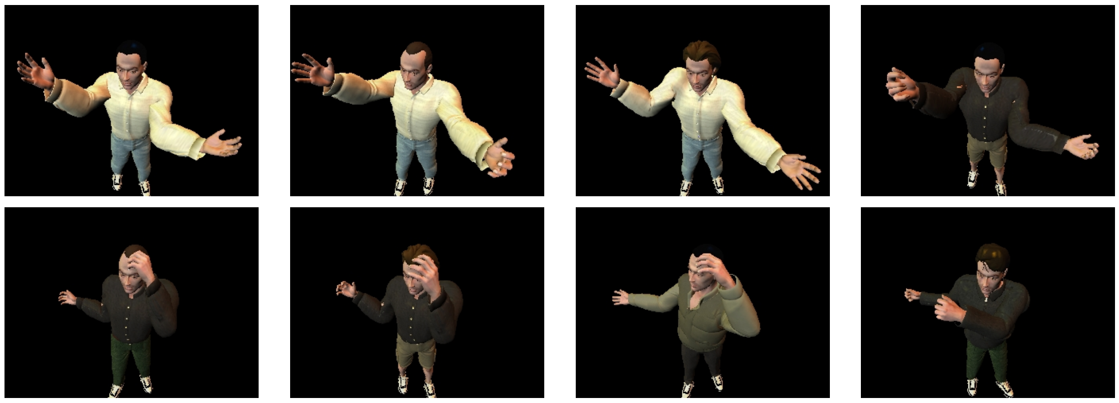


Figure 4-7: Testing on synthetic input. Column 1: test images. Columns 2-4: top 3 matches in H.

two random dimensions of the EDH, and optimized as described in Section 3.4.2, and the projections are combined by the semi-supervised boosting algorithm introduced in Section 3.4.1. In this way we constructed a 1,000-dimensional embedding H.

To get a better understanding of the relationship between independently selecting the dimensions of H with SSC and applying a greedy ensemble learning algorithm in BoostPro, we also measured the TP-FP gap of the selected dimensions. As may be expected, some of the selected features, when considered alone, have very low gap values (as low as .02), nevertheless, they are selected by the boosting since their weighted gap, or equivalently the value of the objective r_m is high.

Figures 4-7 and 4-8 show examples of retrieval by exact NN search in the embedding space H. A more thorough evaluation of the error is reported in the next chapter, where we discuss integration of our pose estimation approach into a tracking framework.

4.6 Discussion

We have presented an example-based approach to articulated pose estimation from a single image. Its main difference from the previously proposed methods is that it does not attempt to build a global model of pose-image relationship, which is notoriously difficult. Instead, we use a large synthetic database to directly learn to detect when the poses underlying two images are similar, and, at the same time, construct an embedding into a space where that similarity is modeled by low L_1 distance between embedded images. The embedding framework and the resulting ability to retrieve similar poses by a simple L_1 search combined with the power of LSH give this approach a critical advantage: the solution to the complex problem of pose estimation becomes very simple and very fast. To our knowledge, no other single-frame pose estimation method that achieves similarly accurate estimates has a comparable speed. These properties make this pose estimation approach well suited as a component in articulated tracking algorithms. In the next chapter we describe two systems in which this is taken advantage of.

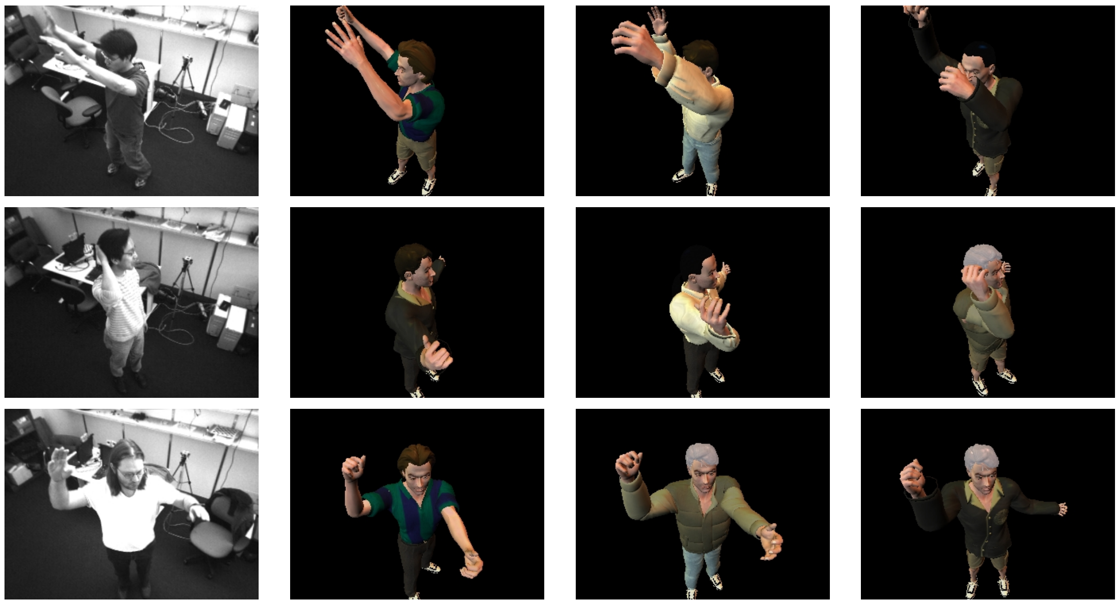


Figure 4-8: Results on real input. Column 1: test images. Columns 2-4: top 3 matches in H