massachusetts institute of technology — artificial intelligence laboratory

Fast Pose Estimation with Parameter Sensitive Hashing

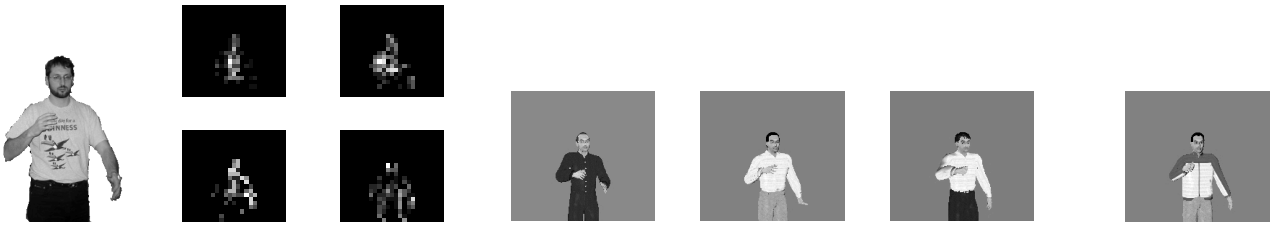
Gregory Shakhnarovich, Paul Viola
and Trevor Darrell

AI Memo 2003-009 April 2003

Abstract

Example-based methods are effective for parameter estimation problems when the underlying system is simple or the dimensionality of the input is low. For complex and high-dimensional problems such as pose estimation, the number of required examples and the computational complexity rapidly becme prohibitively high. We introduce a new algorithm that learns a set of hashing functions that efficiently index examples relevant to a particular estimation task. Our algorithm extends a recently developed method for locality-sensitive hashing, which finds approximate neighbors in time sublinear in the number of examples. This method depends critically on the choice of hash functions; we show how to find the set of hash functions that are optimally relevant to a particular estimation problem. Experiments demonstrate that the resulting algorithm, which we call Parameter-Sensitive Hashing, can rapidly and accurately estimate the articulated pose of human figures from a large database of example images.

[0]Part of this work was done when G.S. and P.V. were with Mitsubishi Electric Research Labs, Cambridge, MA.



(a) Input

(b) A subset of the features (edge direction bins, $\pi/8, 3\pi/8, 5\pi/8, 7\pi/8$ from top left clockwise)

(c) 3 top matches in PSH, left to right

(d) Robust LWR estimate

Figure 1: Pose estimation with parameter-sensitive hashing and local regression.

1. Introduction

Many problems in computer vision can be naturally formulated as parameter estimation problems: given an image or a video sequence $\mathbf{x}$, we estimate the parameters θ of a model describing the scene or the object of interest. Examples include the esimation of the location and pose of a human body, the configuration of the hand, the contraction of muscles in the face, or the rotation of an object. For a given estimation task, it is often possible to collect a large set of examples for which the true values of the underlying parameters of interest are known. This can typically be achieved in either of two ways: by acquiring real data while measuring the "ground truth" (e.g. using a motion-capture system); or by creating synthetic data from a generative model of sufficient realistic quality (e.g. animation software). Example-based learning methods can capitalize on the availability of such data: they infer the parameter values for the input from the known values in similar examples.

In this paper we describe a new algorithm for fast parameter estimation using local models estimated from a large database of examples. Classic methods for example-based learning, such as the k-nearest neighbor rule (k-NN) and locally-weighted regression (LWR), are appealing due to their simplicity and the asymptotically optimal quality of the resulting estimators. However, the prohibitive computational complexity of similarity search in high-dimensional spaces and on very large data sets has made these methods infeasible for many vision problems.

We overcome the problem of computational complexity by using a recently developed algorithm for fast approximate neighbor search, Locality-Sensitive Hashing (LSH)[12]. The training examples are indexed by a number of hash tables, such that the probability of collision is large for examples similar in their parameters and small for dissimilar ones. The query time is $O(dn^{1/(1+\epsilon)})$, where n the number of examples, d the number of features, and ϵ the approximation factor. For many practical problems, such as pose estimation, good results are achieved with ϵ of 1 or 2, which means a speedup factor of 10^3 to 10^4 over an exhaustive search in a database of 10^6 examples.

While LSH provides a technique for quickly finding close neighbors in the input space, these are not necessarily close neighbors in the parameter space. The main contribution of this paper is Parameter-Sensitive Hashing (PSH), an extension of LSH. PSH hashes the data using hash functions sensitive to the similarity in the parameter space, and retrieves approximate nearest neighbors in that space in sublinear time. The key construction is a new feature space that is learned from examples in order to more accurately reflect distances in parameter space. We show how the objective of parameter sensitivity can be formulated in terms of a classification problem, and propose a simple and efficient algorithm for evaluating this objective and selecting parameter-sensitive hash functions. Finally, we use robust locally-weighted regression (LWR) to more accurately estimate parameters using the approximate nearest neighbors. To our knowledge, this is the first use of an LSH technique for local regression.

We apply our framework to an articulated pose estimation problem: estimating the pose of a human upper body from a single image. The remainder of this paper is organized as follows. Prior results are reviewed in Section 2. The PSH algorithm is presented in Section 3. One critical component is an algorithm for constructing efficient hash functions, described in Section 3.1. The articulated pose estimation task is described in Section 4. We conclude with discussion and ideas for future work in Section 5.

2. Background and related work

The problem of estimating the pose of articulated bodies has been of increasing recent interest. In [18] 3D pose is recovered from the 2D projections of a number of known feature points on an articulated body. Efficient algorithms for matching articulated patterns were given in [11, 16]. These approaches assume that detectors are available for specific feature locations. More generally, [14, 15] describe a 'shape context' feature vector to represent general contour shape. In [17], 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 unable to use appearance within a silhouette.

The idea of example-based pose estimation from intensity images was explored in [2], where a hand image was matched to a large database of rendered forms. This work is most similar to ours and in part inspired our approach. However, this approach has difficulty scaling to the large numbers of examples needed for general articulated pose estimation due to the complexity of nearest neighbor search with image-based distance metrics. We approach pose estimation as a local learning task, and exploit recent advances in locality- sensitive hashing to make example-based learning feasible for pose estimation. We review each of these topics in turn.

2.1. Local learning models

The parameter esimation problem can be formulated as estimating the inverse of an unknown generative process for images $\mathbf{x} = f(\theta)$, given a training set $(\mathbf{x}_1, \theta_1), \ldots, (\mathbf{x}_N, \theta_N)$, where distance functions in the parameter space d_θ and in the input space d_X are available. Estimation should minimize the residual in terms of d_θ, which is the relevant similarity measure; however, for a novel input we can measure only d_X between example images. In the context of articulated pose estimation, θ is often a vector of joint angles, and $\mathbf{x}$ is the feature vector computed from an image.

Nearest neighbor (NN) retrieval, and the generalization to k-NN, are among the oldest techniques for such estimation [1]. The k-NN estimate is obtained by averaging the values for the k training examples most similar to the input:

$$\hat{\theta}_{NN} = \frac{1}{k} \sum_{\mathbf{x}_i \in \text{neighborhood}} \theta_i, \tag{1}$$

i.e. the target function is approximated by a constant in each neighborhood defined by k. Properties of k-NN, particularly the consistency and the asymptotically Bayes-optimal risk under many loss functions [8, 7], make it appealing for classification and estimation.

A natural extension to k-NN, in which the neighbors are weighted according to their similarity to the query point, leads to locally-weighted regression (LWR) [5, 3]: the unknown function is assumed to be approximated locally (within any small region) by a function from a particular model class $g(\mathbf{x}; \beta)$. The parameters β are chosen to optimize the weighted learning criterion in the test input $\mathbf{x}_0$,

$$\beta^* = \operatorname*{argmin}_\beta \sum_{i=1}^{N} d_\theta \left(g(\mathbf{x}_i, \beta), \theta_i \right) K \left(d_X(\mathbf{x}_i, \mathbf{x}_0) \right), \tag{2}$$

where K is the *kernel* function that determines the weight falloff with increasing distance from the query point; the summation is over the examples in the neighborhood.

In *robust* LWR [4], the influence of outliers on the final estimate is diminished through a short iterative process. This is a particularly useful safeguard when dealing with approximate neighborhoods, and with indirect similarity measures. In a nutshell, after the model is fit, the neighborhood points are reweighted so that points with higher residual w.r.t. the fitted values become less influential.

There are two major obstacles in the straighforward application of example-based methods to the problem of parameter estimation from images. The first obstacle is the computational complexity of the NN search, particularly in the high-dimensional spaces often encountered in machine vision tasks. Using approximate NN algorithms such as LSH may overcome this obstacle. While the idea of using LSH in a similar task has been mentioned by others [14, 2], to the best of our knowledge no experiments actually using it have been performed.

The second obstacle is the reliance on d_X, the image/feature metric, which is assumed to be positively correlated with d_θ. We will show how to explicitly select a feature space in which d_X approximates d_θ, without an explicit global model of this relationship. The approximate nearest neighbors retrieved using this space are of much higher quality that those retrieved using more generic feature spaces.

2.2. Locality-Sensitive Hashing

The following relaxed version of near neigbor search, called (r, ϵ)-NN, can be solved in sublinear time by LSH [12]: if for a query point $\mathbf{u}$ there exists a point $\mathbf{v}$ such that $d(\mathbf{u}, \mathbf{v}) < r$, then a point $\mathbf{v}'$ is returned such that $d(\mathbf{u}, \mathbf{v}') < (1+\epsilon)r = R$. Otherwise, the absence of such point is reported. Before we summarize the LSH algorithm, we must define what is meant by locality-sensitive.

Given the example space X, a family $\mathcal{H}$ of functions is called *locality-sensitive*, or more specifically $(r, r(1+\epsilon), p_1, p_2)$-sensitive, if for any $\mathbf{u}, \mathbf{v} \in X$,

$$\text{if } d(\mathbf{u}, \mathbf{v}) \ \leq \ r \text{ then } \Pr_{\mathcal{H}}\left(h(\mathbf{u}) = h(\mathbf{v})\right) \geq p_1, \tag{3}$$

$$\text{if } d(\mathbf{u}, \mathbf{v}) \ > \ (1+\epsilon)r \text{ then } \Pr_{\mathcal{H}}\left(h(\mathbf{u}) = h(\mathbf{v})\right) \leq p_2, \tag{4}$$

where $\Pr_{\mathcal{H}}$ is the probability with respect to a random choice of $h \in \mathcal{H}$. We will assume, w.l.o.g., that every $h \in \mathcal{H}$ is binary.

A *k-bit locality-sensitive hash function* (LSHF) g

$$g(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_k(\mathbf{x})]^T \tag{5}$$

constructs a hash key by concatenating the bits computed by a randomly selected set of $h_1, \ldots, h_k$. Note that the probability of collision for similar points is at least $1-(1-p_1)^k$, while for dissimilar points it is at most p_2^k. A useful LSHF must have $p_1 > p_2$ and $p_1 > 1/2$.

The algorithm preprocesses the data by storing it independently into l hash tables indexed by $g_1, \ldots, g_l$. For a query point $\mathbf{x}$, the candidate examples are found in l corresponding buckets in the hash tables, and the exhaustive NN search is only done on those examples; if the algorithm succeeds, an ϵ-NN of $\mathbf{x}$ is among them.

The values of l and k affect both the precision and the efficiency of LSH. A large l increases the probability of finding good examples, but also the potential number of candidate examples. A large k speeds up the search by reducing the number of collisions, but also increases the probability of a miss. Suppose that our goal is to search exhaustively at most B examples for each query; then setting

$$k = \log_{1/p_2}\left(\frac{n}{B}\right), \quad l = \left(\frac{n}{B}\right)^{\frac{\log(1/p_1)}{\log(1/p_2)}} \tag{6}$$

ensures [12] that LSH will succeed with high probability.

The construction of an efficient set of LSHFs (with high p_1 and low p_2) is obviously critical to the success of the algorithm. In Section 3.1 we suggest a learning algorithm for constructing such a set for parameter estimation.

3. Estimation with Parameter-Sensitive Hashing

Let $(\mathbf{x}_1, \theta_1), \ldots, (\mathbf{x}_N, \theta_N)$ be the training examples with their associated parameter values. We assume that an example is represented by a D-dimensional feature vector $\mathbf{x} = [x^1, \ldots, x^D]$ where $x^j = \phi_j(\mathbf{x})$ is computed by a scalar-valued function, such as a filter response at a certain location or a bin value in an edge direction histogram in a certain region of the image.

We assume that the following criteria are met by the estimation problem:

1. A distance function d_θ is given which measures similarity between parameter vectors.
2. A radius R in the parameter space is given such that θ_1, θ_2 are considered similar iff $d_\theta(\theta_1, \theta_2) < R$.
3. The training examples are representative of the problem space, i.e. for a randomly drawn example there exists, with high probability, an example with similar parameter values.
4. The process that generates the examples is unbiased, or it is possible to correct for such bias.

The distance function and the similarity threshold are dependent on the particular task, and often reflect perceptual similarities between the scenes or objects. Without these, even evaluation of the estimation results is impossible.

The third assumption may appear a bit vague, and in fact its precise meaning depends on the nature of the problem. If we control the example generation process, the required property can be achieved by "filling" the space, storing an example in every node on an R-grid in parameter space. This becomes infeasible very quickly as the dimension of θ increases. Alternatively, it has been observed or conjectured [13, 19] that images of many real-world phenomena do not fill the space uniformly, but rather belong to an intrinsically low-dimensional subspace, or manifold, and densely covering that subspace is enough to ensure the third property.

The last assumption is perhaps the most limiting. It implies that there are no significant sources of variation in the examples besides the variation in the parameters, or that the contribution of such sources can be accounted for. This is possible in many vision problems, either explicitly, by normalizing the examples, or implicitly, e.g. by using features invariant with respect to the "nuisance" parameters.

3.1. Parameter-sensitive hash functions

Let $p_1(h)$ and $p_2(h)$ be the probabilities of collision for similar/different examples for a given hash function. Clearly,

$$p_1 \geq \min_{h \in \mathcal{H}} p_1(h),\ p_2 \leq \max_{h \in \mathcal{H}} p_2(h). \tag{7}$$

We know (Section 2.2) that a family of hash functions is useful when p_2 is low and p_1 is high, and we are motivated to construct $\mathcal{H}$ from functions that achieve this. In [12] quantities like $p_1(h)$ are derived in the simplified case where the goal is to find near neighbors in the input (or image) space. For the parameter estimation task, where the goal is to find near neighbors in the unknown parameter space, an analytic derivation of $p_1(h)$ and $p_2(h)$ is infeasible since h is a measurement in the image domain.

However, we can show that $p_1(h)$ and $p_2(h)$ have an intuitive interpretation in the context of the following classification problem. Let us assign to each possible pair of examples $(\mathbf{x}_i, \mathbf{x}_j)$ the label

$$y_{ij} = \begin{cases} +1 \text{ if } d_\theta(\theta_i, \theta_j) < r, \\ -1 \text{ if } d_\theta(\theta_i, \theta_j) > R, \\ \text{not defined otherwise,} \end{cases} \tag{8}$$

where $r = R/(1+\epsilon)$. Note that we do not define the label for the "gray area" of similarity between r and R, in order to conform to Eq. (3).

We can now formulate a classification task consisting of predicting the label for a pair of examples. A binary hash function h either has a collision $h(\mathbf{x}_i) = h(\mathbf{x}_j)$ or not; we define the label predicted by h as

$$\hat{y}_h(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} +1 \text{ if } h(\mathbf{x}_i) = h(\mathbf{x}_j), \\ -1 \text{ otherwise.} \end{cases} \tag{9}$$

It is now clear that, when $\hat{y}$ is interpreted as a classifier, $p_2(h)$ is the probability of a false positive, i.e. $\Pr(\hat{y}_{ij} = +1 | y_{ij} = -1)$, and similarly $1 - p_1(h)$ is the probability of a false negative. Our objective therefore is to find h's with high prediction accuracy. This can be done by evaluating h on a large set of paired examples for which true labels can be computed. Such a *paired problem* set can be built from our training set, since we know d_θ, r, R. We should be careful about two things when constructing the paired problem. First, we must not include pairs with similarity within the "gray area" between r and R. Second, we should take into account the asymmetry of the classification task: there are many more negative examples among possible pairs than there are positive. Consequently, in order to represent the negative examples appropriately, we must include many more of them in the paired problem.

The exact nature of the hash functions h will affect the feature selection algorithm. Here we offer an efficient algorithm for simple threshold functions

$$h_{\phi,T}(\mathbf{x}) = \begin{cases} +1 \text{ if } \phi(\mathbf{x}) \geq T, \\ -1 \text{ otherwise.} \end{cases}$$

where $\phi(\mathbf{x})$ is some continuous image function and T is a threshold. The classification implied by this hash function is therefore $\hat{y}_h(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i)h(\mathbf{x}_j)$.

The search for effective hash functions is now a search for a set of $h_{\phi,T}()$ each yielding high accuracy on the paired problem. For a given ϕ, the optimal T can be found in two passes over the paired training set (see Figure 2).

3.2. Similarity search

Once an effective set of hashing functions is found, LSH is used to query the database rapidly. A query is performed by computing the hash buckets $g_1(\mathbf{x}_0), \ldots, g_l(\mathbf{x}_0)$. Some of the buckets, if not all, may be empty; in the latter case, the algorithm terminates in failure mode.

```
Given: Feature φ
Given: Pairs X̃ = (x_{i_n}, x_{j_n}, y_n)_{n=1}^N.
  Start with an empty array A.
  T_p := number of positive pairs
  T_n := number of negatives pairs
  for n = 1 to N do
    v_1 := φ(x_{i_n}), v_2 = φ(x_{j_n})
    l_1 := 1 if v_{i_n} > v_{j_n}, 0 otherwise
    l_2 := −l_1
    A := A ∪ {< v_1, l_1, n >, < v_2, l_2, n >}
  end for
  At this point A has 2N elements.  Each paired example is represented twice.
  Sort A by the values of v
  S_p := S_n := 0
  c_best := T_n
  for k = 1 to 2N do
    Let < v, l, n >= A[k]
    if y_n = +1 then
      S_p := S_p − l
    else if y_n = −1 then
      S_n := S_n − l
    end if
    c := (T_n − S_n) + S_p
    if c < c_best then
      c_best := c; T_best := v
    end if
  end for
```

Figure 2: Algorithm for PSHF selection. Intuitively, the algorithm tries all possible thresholds and counts the number of negative examples that are assigned the same hash value and positives that are assigned different values. Since the exmaples are sorted by feature value, these quantities can be updated with little work. The threshold T_{best} is the one that minimizes their sum.

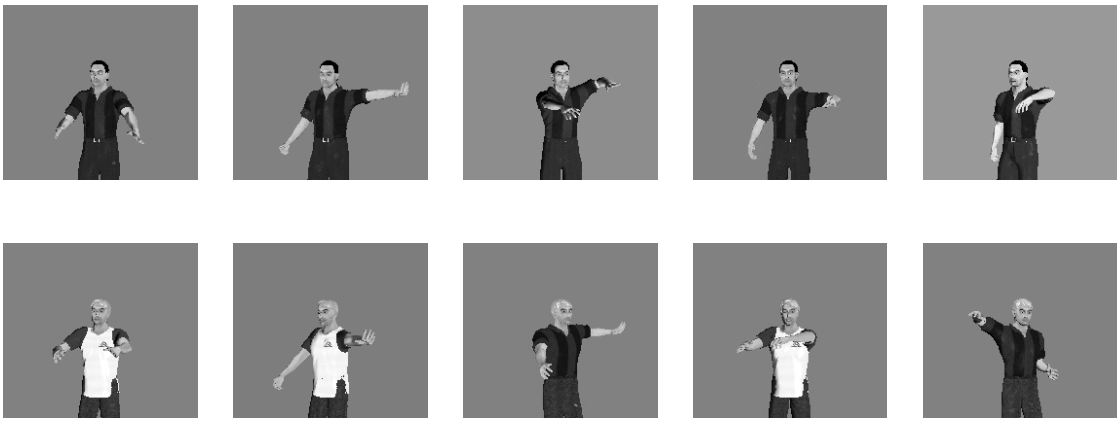
Let X' be the union of the non-empty buckets. The number M of examples in X' can still be quite large, especially if the number of hash tables is large and/or the number of hash bits k is small, but $M \ll N$ with high probability. X' is then exhaustively searched to produce the K ϵ-NN $\mathbf{x}'_1, \ldots, \mathbf{x}'_K$, ordered by increasing $d_X(\mathbf{x}'_i, \mathbf{x}_0)$, with parameters $\theta'_1, \ldots, \theta'_K$. These are the data points on which we will perform local regression.

3.3. Local regression

The simplest way to proceed is to return θ'_1 as the answer. There are two problems with this. First, even if LSH succeeds, θ'_1 can still be up to R away from the true parameter of the input, θ_0 (e.g., see Figure 6). In most practical situations, the R for which it is feasible to satisfy the representativeness property mentioend above is too large to make this an acceptable solution. A second problem is caused by our inability to directly measure $d_\theta(\theta_0, \theta)$. We must rely on the properties of LSHF, and on the monotonicity of d_X with respect to d_θ, which are usually not perfect. Therefore, we need a way to correct the estimate once the approximate neighborhood has been found.

A possible way of achieving this is by using the k-NN estimate as a starting point of a gradient descent search [2]. Alternatively, active learning can be used to refine the "map" of the neighborhood [6]. Both approaches, however, require an explicit generative model of $p(\mathbf{x}|\theta)$, or "oracle", which for a given value of θ generates an example to be matched to $\mathbf{x}_0$. While in some cases it is possible (i.e. animation software which would render objects with a given pose), we would like to avoid such a limitation.

Instead, we use robust LWR. Since we expect the number of neighbors to be small, we use low-order polynomials (constant or linear) to avoid overfitting. The parameters of LWR in Eq. 2, e.g. the degree of g (0 or 1) and the kernel bandwidth, as well as the number of iterations of reweighting, are chosen based on validation set performance.



(a) $d_\theta = .21$, $d_X = .8127$ $d_I = .271$

(b) $d_\theta = .505$, $d_X = .9674$ $d_I = .261$

(c) $d_\theta = 1.001$ $d_X = .8079$ $d_I = .112$

(d) $d_\theta = 3$ $d_X = .9617$ $d_I = .265$

(e) $d_\theta = 6.32$ $d_X = 1.1086$ $d_I = .126$

Figure 3: Examples of positive and negative examples from the paired problem. For each pair, the true label is $+1$ iff $d_\theta < 0.25$, i.e. only the leftmost example is positive. Distances d_X in feature space and d_I (normalized pixel-wise L_2 distance) are given for reference.

4. Pose estimation with PSH

We apply our algorithm to the problem of recovering the articulated pose of a human upper body. The model has 13 degrees of freedom: one DOF for orientation, namely the rotation angle of the torso around the vertical axis, and 12 DOFs in rotational joints (2 in each collar, 3 in each shoulder, and 1 in each elbow). We do not assume constant illumination or fixed poses for other body parts in the upper body (head and hands), and therefore need to represent the variation in these and other nuisance parameters, such as clothing and hair style, in our training set.

For this application, it is important to separate the problem of object detection from that of pose estimation. Given simple backgrounds and a stationary camera, body detection and localization is not difficult. In these experiments, it is assumed that the body has been segmented, scaled, and centered in the image. For more difficult scenarios, a more complex object detection system may be required.

Input images are represented in our experiments by multi-scale edge direction histograms, obtained in the following way : edges are detected using the Sobel operator and each pixel is classified into one of four direction bins ($\pi/8, 3\pi/8, 5\pi/8, 7\pi/8$). Then, the histograms of direction bins are computed within sliding square windows of varying sizes (8,16, and 32 pixels) placed at multiple locations in the image. The feature space consists of the concatenated values of all of the histograms. We chose this representation, often used in image analysis [10], because it is largely invariant to some of the nuisance parameters with respect to pose, such as illumination and color. Figure 1(b) illustrates a subset of the features, namely half of the 8×8 histogram bins.

The training set consisted of 150,000 images rendered from a humanoid model using POSER [9], with pose parameter values sampled independently and uniformly within certain anatomically feasible ranges; the torso orientation is constrained to the range $[-40^o, 40^o]$. Each training image is 180×200 pixels. In our pose estimation experiments, the angles are constrained to $[-\pi, \pi]$, so we use the sine distance defined as

$$d_\theta(\theta_1, \theta_2) = \sum_{i=1}^{m} 1 - \cos(\theta_1^i - \theta_2^i) \tag{10}$$

where m is the dimension of the parameter space (number of joint angles), and θ_j^i is the i-th component of θ_j. We found that this distance function usually reflects our perception of pose similarity (see Figure 3 for examples).

Figure 4 shows the distribution of pairwise distances in the training set. After examining large numbers of images corresponding to poses with various distaces, we set $r = 0.25$. The LSH queries should therefore return examples within $R = 0.5$

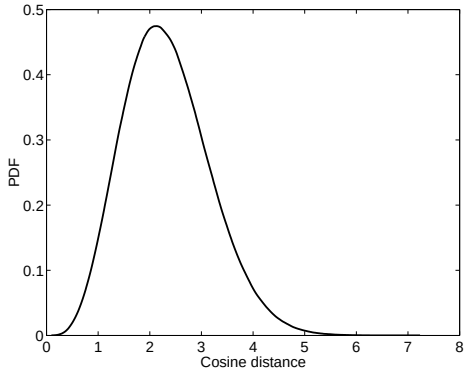


Figure 4: Distribution of d_θ in the upper body training set; $R = 0.5$.

Model	Mean error (d_θ)			
	$k = 3$	$k = 7$	$k = 12$	$k = 25$
k-NN	1.001	.856	.801	.791
Linear LWR	.999	.854	.800	.786
Constant LWR	.995	.854	.780	.788
Robust linear LWR	1.180	.887	.799	.740
Robust constant LWR	1.096	.860	.783	.749

Table 1: Average error for different local models; not shown is the error of 1-NN, 2.046.

with high probability. Figure 3 shows 5 of 1,775,000 paired examples used by PSH to select 269 out of 11,270 features. All features achieve an accuracy better than 0.465 on the paired problem; this threshold was set to balance the number of features and their accuracy. Based on Eq. 6, PSH was implemented via 200 hash tables using 18-bit hash functions.

To quantitatively evaluate the algorithm's performance, we tested it on 500 synthetic images, generated from the same model. Table 1 shows the average error between the true and estimated poses obtained with different local learning models. Recall that $d_\theta < 0.5$ is considered a correct result. On average the PSH returned 5000 candidates for NN; in almost all cases, the true nearest neighbors under d_X were also the top PSH candidates.

We also tested the algorithm on 800 images of a real person; images were processed by a simple segmentation and alignment program. Figure 5 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 vizualization of the pose estimated with robust linear LWR on 12-NN as approximated by PSH; we used Gaussian kernel with the bandwidth set to the d_X distance to the 12-th neighbor. In some cases, there is a visible improvement versus the 1-NN estimate based on the top match in PSH. The number of candidates in PSH was significantly lower than for the synthetic images - about 2000, or 1.3% of the database; this can be explained by the fact that two synthetic images are more likely to have exactly equal values for many features. It takes an unoptimized Matlab program less than 2 seconds to produce the pose estimate (not including the rendering). This is a dramatic improvement over searching the entire database for the exact NN, which takes more than 2 minutes per query, and in most cases produces the same top matches as the PSH.

Lacking ground truth for these images, we rely on visual inspection of the pose estimate. For most of the examples the pose estimate was accurate; on some examples it failed to various extents. Figures 5 and 6 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 good fit. We believe that there are three main sources of failure: significant mismatch between d_θ and d_X, imperfect segmentation and alignment, and the limitations of the training set, in terms of coverage and representativeness of the problem domain.

Figure 5: Examples of upper body pose estimation (Section 4). Top row: input images. Middle row: top PSH match. Bottom row: robust LWR (RLWR) estimate based on 12 NN.

Figure 6: More examples, including typical “errors”. In the leftmost column, the gross error in the top match is corrected by LWR. The rightmost two columns show various types of errors.

5. Summary and Conclusions

We present an algorithm that uses new hashing-based search techniques to rapidly find relevant examples in a large database of image data, and estimates the parameters for the input using a local model learned from those examples. Experiments show that our estimation method, based on parameter-sensitive hashing and robust locally-weighted learning, is successful on the task of articulated pose estimation from static input. These experiments also demonstrate the usefulness of synthetically created data for learning and estimation.

In addition to the use of local regression to refine the estimate, our work differs from that of others, e.g. [2, 14], in that it allows accurate estimation when examining only a fraction of a dataset. The running time of our algorithm is sublinear; in our experiments we observed a speedup of almost 2 orders of magnitude relative to the exhaustive exact nearest-neighbor search, reducing the time to estimate pose from an image from minutes to under 2 seconds without adversely affecting the accuracy. We expect an optimized version of the system to run at real time speed. This has the potential of turning infeasible example-based estimation methods into attractive for such tasks.

There are many interesting questions that remain open. The learning algorithm, presented in Section 3.1, implicitly assumes independence between the features; we are exploring more sophisticated feature selection methods that would account for possible dependencies. Our experiments so far have involved estimating pose from a single frame, ignoring temporal constraints on human motion. Finally, as we mentioned earlier, the presented framework is not specific to pose; we intend to investigate its use in other parameter estimation tasks.

References

[1] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, August 1992.

[2] V. Athitsos and S. Sclaroff. An appearance-based framework for 3d hand shape classification and camera viewpoint estimation. In *Proceedings of the Int. Conf. on Automatic Face & Gesture Recognition*, pages 45–50, Washington, DC, 2002.

[3] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[4] W. S. Cleveland. Robust locally weighted regression and smoothing scatter plots. *Journal of American Statistical Association*, 74(368):829–836, 1979.

[5] W. S. Cleveland and S. J. Delvin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of American Statistical Association*, 83(403):596–610, 1988.

[6] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *J. Artificial Intelligence Research*, 4:129–145, 1996.

[7] T. M. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:21–27, January 1968.

[8] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, January 1967.

[9] Curious Labs, Inc., Santa Cruz, CA. *Poser 5 - Reference Manual*, 2002.

[10] A. Evans, N. Thacker, and J. Mayhew. The use of geometric histograms for model-based object recognition. In *British Machine Vision Conference*, pages 429–438, 1993.

[11] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 66–75, Los Alamitos, June 13–15 2000. IEEE.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*, pages 518–529, San Francisco, September 1999. Morgan Kaufmann.

[13] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

[14] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 723–730, Lihue, HI, 2001.

[15] G. Mori and J. Malik. Estimating Human Body Configurations using Shape Context Matching. In *Proceedings of European Conference on Computer Vision*, 2002.

[16] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *Proceedings of European Conference on Computer Vision*, Copenhagen, Denmark, 2002.

[17] R. Rosales and S. Sclaroff. Specialized mappings and the estimation of body pose from a single image. In *IEEE Human Motion Workshop*, pages 19–24, Austin, TX, 2000.

[18] C. J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80(3):349–363, December 2000.

[19] Y. Wu, J. Y. Lin, and T. S. Huang. Capturing natural hand articulation. In *Proceedings of the International Conference on Computer Vision*, pages 426–432, Vancouver, BC, 2001.