Fast Pose Estimation with Parameter-Sensitive Hashing

Gregory Shakhnarovich[1*] gregory@ai.mit.edu
Paul Viola[2] viola@microsoft.com
Trevor Darrell[1] trevor@ai.mit.edu

[1]Computer Science and Artificial Intelligence Lab, MIT
Cambridge, MA 02139

[2]Microsoft Research
Redmond, WA 98052

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 become prohibitively high. We introduce a new algorithm that learns a set of hashing functions that efficiently index examples in a way relevant to a particular estimation task. Our algorithm extends locality-sensitive hashing, a recently developed method to find 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.

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 estimation of the configuration of an articulated body, the contraction of muscles in the face, or the orientation of a rigid object. Example-based estimation methods capitalize on the availability of a large set of examples for which the parameter values are known: they infer the parameter values for the input from the known values in similar examples. This does not require modeling the global structure of the input/parameter relationship, which is only assumed to be sufficiently smooth to make such inference meaningful.

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 asymptotic optimality of the resulting estimators. However, the computational complexity of similarity search, used by these methods, in high-dimensional spaces and on very large data sets has made them infeasible for many vision problems.

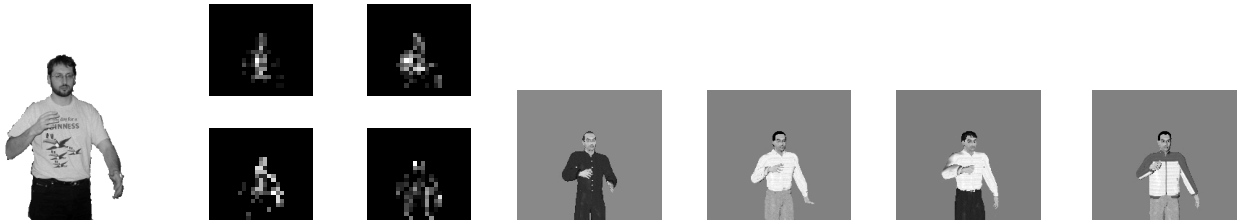
In this paper we describe a new example-based algorithm for fast parameter estimation using local models, which are dynamically built for each new input image. We overcome the problem of computational complexity with a recently developed algorithm for fast approximate similarity search, Locality-Sensitive Hashing (LSH) [11]. 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. For practical problems, such as pose estimation, good results can be achieved with a speedup factor of 10^3 to 10^4 over an exhaustive search in a database as large as 10^6 examples.

What one really wants is to base the estimate on examples similar to the input in their parameter values as well as in the input space. Note, however, that while LSH provides a technique for quickly finding close neighbors in the input space, these are not necessarily close neighbors in the parameter space. An exact solution for this task would require knowledge of the parameter values for the input - precisely the problem one needs to solve!

The main contribution of this paper is Parameter-Sensitive Hashing (PSH), an extension of LSH. PSH uses hash functions sensitive to the similarity in the parameter space, and retrieves in sublinear time approximate nearest neighbors of the input with respect to parameter values as well as the features. The key construction is a new binary feature space that is learned from examples in order to more accurately reflect the proximity in parameter space. We show how the objective of parameter sensitivity can be for-

*Part of this research was performed while first two authors were at Mitsubishi Electric Research Labs, Cambridge, MA. Their support is gratefully acknowledged.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE



(a) Input (b) A subset of the features (c) 3 top matches in PSH, left to right (d) Robust LWR

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

mulated in terms of a classification problem, and propose a simple and efficient algorithm for evaluating this objective and selecting parameter-sensitive hash functions. Finally, the goal estimate is produced by *robust LWR* which uses the approximate neighbors found by PSH to dynamically build a simple model of the neighborhood of the input. Our approach is illustrated in Figure 1. To our knowledge, this is the first use of an LSH-based technique with local regression.

The remainder of this paper is organized as follows. Previous work is reviewed in Section 2. The PSH algorithm is presented in Section 3 (the algorithm for constructing efficient hash functions is described in Section 3.1). We evaluate our framework on an articulated pose estimation problem: estimating the pose of a human upper body from a single image. The details of the task and our experiments are described in Section 4. We conclude and discuss some open questions in Section 5.

2. Background and previous work

The body of literature on object parameter estimation from a single image, and in particular on estimating the pose of articulated bodies, is very large, and space constraints force us to mention only work most related to our approach.

In [17] 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 [9, 15]. These approaches assume that detectors are available for specific feature locations, and that a global model of the articulation is available. In [14] a 'shape context' feature vector is used to represent general contour shape. In [16], 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.

Finally, in [1] 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. 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.

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. Example-based estimation

The task of example-based parameter estimation in vision can be formulated as follows. Input, which consists of image features (e.g. edge map, vector of responses of a filter set, or edge direction histograms) computed on the original image, is assumed to be generated by an unknown parametric process $\mathbf{x} = f(\theta)$ (e.g., θ is a vector of joint angles in the articulated pose context. A training set of labeled examples $(\mathbf{x}_1, \theta_1), \ldots, (\mathbf{x}_N, \theta_N)$ is provided. One must estimate θ_0 as the inverse of f for a novel input $\mathbf{x}_0$. The objective is to minimize the residual in terms of the distance (similarity measure) d_θ in the parameter space.

Methods based on nearest neighbors (NN) are among the oldest techniques for such estimation. The *k-NN estimate* [7] is obtained by averaging the values for the k training examples most similar to the input:

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

i.e. the target function is approximated by a constant in each neighborhood defined by k. This estimate is known to be consistent, and to asymptotically achieve Bayes-optimal risk under many loss functions [7]. Note that similarity (neighborhood) is defined in terms of the distance d_X in the input space.

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, 2]: the target function is 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

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

criterion in the test input $\mathbf{x}_0$,

$$\beta^* = \underset{\beta}{\operatorname{argmin}} \sum_{\mathbf{x}_i \in \text{neighborhood}} d_\theta\left(g(\mathbf{x}_i;\beta),\theta_i\right) K\left(d_X(\mathbf{x}_i,\mathbf{x}_0)\right), \tag{2}$$

where K is the *kernel* function that determines the weight falloff with increasing distance from the query point.

In *robust* LWR [4], the influence of outliers is diminished through a short iterative process. In each iteration after the model is fit, the neighborhood points are re-weighted so that points with higher residual w.r.t. the fitted values become less influential.

There are two major problems with the straightforward application of example-based methods to parameter estimation in vision. The first is the computational complexity of the existing NN search algorithms, particularly in the high-dimensional spaces often encountered in vision tasks. Using fast approximate NN algorithms may overcome this obstacle. The idea of using approximate NN has been mentioned in previous work for object or texture classification [3, 10], and for some estimation tasks [13, 1]. However to our knowledge no experiments using recent algorithms for estimation tasks have been conducted.

The second problem, not immediately solved by adopting an efficient similarity search algorithm, is the reliance of the search on the input space metric d_X, without explicitly taking into account d_θ. We will show how to explicitly select a feature subspace in which d_X approximates d_θ, without an explicit global model of this relationship. The approximate NN in this space are of much higher relevance that those retrieved using distance in the original feature spaces.

2.2. Locality-Sensitive Hashing

The following problem, called (r,ϵ)-NN, can be solved in sublinear time by LSH [11]: if for a query point $\mathbf{u}$ there exists a training point $\mathbf{v}$ such that $d(\mathbf{u},\mathbf{v}) \le r$, then (with high probability) a point $\mathbf{v}'$ is returned such that $d(\mathbf{u},\mathbf{v}') \le (1+\epsilon)r$. Otherwise, the absence of such point is reported. We shall now define the term "locality-sensitive" and summarize the LSH algorithm.

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

$$\begin{aligned} \text{if } d(\mathbf{u},\mathbf{v}) \le r \text{ then } \Pr_{\mathcal{H}}\left(h(\mathbf{u}) = h(\mathbf{v})\right) \ge p_1, \\ \text{if } d(\mathbf{u},\mathbf{v}) > (1+\epsilon)r \text{ then } \Pr_{\mathcal{H}}\left(h(\mathbf{u}) = h(\mathbf{v})\right) \le p_2, \end{aligned} \tag{3}$$

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 valued.

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

$$g(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_k(\mathbf{x})]^T \tag{4}$$

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 $p_1 > 1/2$.

In the preprocessing stage of LSH, each training example is entered into l hash tables indexed by independently constructed $g_1,\ldots,g_l$. For a query point $\mathbf{x}_0$, the exhaustive search is only carried out among the examples in the union of l hash buckets indexed by $\mathbf{x}_0$. If the algorithm succeeds, these candidates include an (r,ϵ)-NN of $\mathbf{x}_0$.

The values of l and k affect both the precision and the efficiency of LSH. A large l increases the probability of success, but also the potential number of candidate examples (and thus the running time). 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

$$k = \log_{1/p_2}\left(\frac{N}{B}\right), \quad l = \left(\frac{N}{B}\right)^{\frac{\log(1/p_1)}{\log(1/p_2)}} \tag{5}$$

ensures [11] that LSH will succeed with high probability. Its expected query time is $O\left(dN^{1/(1+\epsilon)}\right)$, which translates into a factor 1000 speedup compared to exhaustive exact search for $N = 10^6, \epsilon = 1$.

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 the next section we develop 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. An example is represented by a feature vector $\mathbf{x} = [x^1,\ldots,x^D]$ where x^j is computed by a scalar-valued function ϕ_j on the input image, such as a filter response at a certain location or a bin count in edge direction histogram in a certain region. We assume the following:

1. A distance function d_θ is given which measures similarity between parameter vectors, and a radius R in the parameter space is given such that θ_1,θ_2 are considered *similar* iff $d_\theta(\theta_1,\theta_2) < R$.
2. 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.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

3. 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.

The second 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, we can attempt to "fill" 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 often observed or conjectured [12, 18] that images of many real-world phenomena do not fill the space uniformly, but rather belong to an intrinsically low-dimensional manifold, and densely covering that manifold is enough to ensure this property.

The last assumption 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. While perhaps limiting, this is possible to comply with 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

For a hash function h, let $p_1(h)$ and $p_2(h)$ be the probabilities of collision for similar/different examples. Recall (Section 2.2) that a family of hash functions $\mathcal{H}$ is useful when, averaged over $h \in \mathcal{H}$, $p_2(h)$ is low and $p_1(h)$ is high. In [11] quantities like $p_1(h)$ are derived for the task of finding neighbors in the input space. For the parameter estimation task, where the goal is to find neighbors in the unknown parameter space, analytic derivation of $p_1(h)$ and $p_2(h)$ is infeasible since h is a measurement in the input (not parameter) 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

$$y_{ij} = \begin{cases} +1 \text{ if } d_\theta(\theta_i, \theta_j) < r, \\ -1 \text{ if } d_\theta(\theta_i, \theta_j) > R, \\ \text{not defined otherwise,} \end{cases} \tag{6}$$

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 related to these labels. A binary hash function h either has a collision $h(\mathbf{x}_i) = h(\mathbf{x}_j)$ or not; we say that h predicts the label

$$\hat{y}_h(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} +1 \text{ if } h(\mathbf{x}_i) = h(\mathbf{x}_j) \quad \text{(collision)}, \\ -1 \text{ otherwise.} \end{cases} \tag{7}$$

Thus, when h is interpreted as a classifier, $p_2(h)$ is the probability of a false positive $\Pr(\hat{y}_{ij} = +1 | y_{ij} = -1)$, 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 consider h which are decision stumps:

$$h_{\phi,T}(\mathbf{x}) = \begin{cases} +1 \text{ if } \phi(\mathbf{x}) \geq T, \\ -1 \text{ otherwise.} \end{cases} \tag{8}$$

where $\phi(\mathbf{x})$ is a real-valued image function and T is a threshold. Figure 2 shows an algorithm which for a given ϕ finds the optimal T in two passes over the paired training set. Intuitively, it tries all possible distinct thresholds and counts the number of negative examples that are assigned the same hash value and positives that are assigned different values. Since examples are sorted by feature value, these quantities can be updated with little work. The threshold T_{best} is the one that minimizes their sum.

The family $\mathcal{H}$ of parameter-sensitive hash functions can now be constructed by selecting only $h_{\phi,T}$ for which $p_1(h_{\phi,T})$ and $p_2(h_{\phi,T})$ evaluated on the paired problem satisfy the desired thresholds.

3.2. Similarity search

After $\mathcal{H}$ is selected, we project the data on only those feature dimensions ϕ for which $h_{\phi,T} \in \mathcal{H}$, obtain binary representation for our data by applying (8), select k and l based on (5) and build the l hash tables. For an unlabeled input, LSH is used to query the database rapidly, and finds the union of the l hash buckets, $X' = \bigcup_{j=1}^{l} g_j(\mathbf{x}_0)$. Let M be the number of distinct points in X'; with high probability $M \ll N$ (if $M = 0$ the algorithms terminates in failure mode). X' is exhaustively searched to produce the K (r, ϵ)-NN $\mathbf{x}'_1, \ldots, \mathbf{x}'_K$, ordered by increasing $d_X(\mathbf{x}'_i, \mathbf{x}_0)$, with parameters $\theta'_1, \ldots, \theta'_K$. The estimate is based on these points, which with high probability, belong to an approximate neighborhood of $\mathbf{x}_0$ both in the parameter and in the input spaces.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

```
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 negative 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 evaluation in the case of decision stumps (see Section 3).

3.3. Local regression

The simplest way to proceed is to return θ'_1 as the answer. There are two problems with this. First θ'_1 can be up to R away from the true parameter of the input, θ_0. Often, the R for which it is feasible to satisfy the representativeness property mentioned above is too large to make this an acceptable solution (see Figure 5 for examples). The second problem is caused by our inability to directly measure $d_\theta(\theta_0, \theta)$; the search relies on the properties of LSHF, and on the monotonicity of d_X with respect to d_θ, which are usually not perfect. We need a robust estimate based on the approximate neighborhood found by PSH.

A possible way of achieving this is by using the k-NN estimate as a starting point of a gradient descent search [1]. 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 an "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 (e.g. animation software which would render objects with a given pose), we would like to avoid such a limitation.

Instead, we use robust LWR. to avoid overfitting, since we expect the number of neighbors to be small, we consider constant or linear model, which can be easily fit with weighted least-squares. The parameters, e.g. the model order and the kernel bandwidth, as well as the number of iterations of re-weighting, can be chosen based on validation set.

4. Pose estimation with PSH

We applied 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 clavicle, 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 the experiments reported here, it is assumed that the body has been segmented from background, scaled, and centered in the image. For more difficult scenarios, a more complex object detection system may be required.

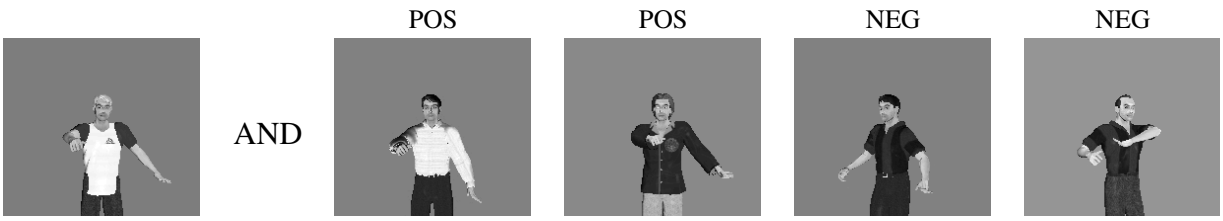
Input images are represented in our experiments by *multi-scale edge direction histograms*. Edges are detected using the Sobel operator and each edge pixel is classified into one of four direction bins: $\pi/8, 3\pi/8, 5\pi/8, 7\pi/8$. Then, the histograms of direction bins are computed within sliding square windows of varying sizes (8, 16, 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 and retrieval, 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 [8], with parameter values sampled independently and uniformly within anatomically feasible ranges; the torso orientation is constrained to the range $[-40^o, 40^o]$. Each training image is 180×200 pixels. In our model, all angles are constrained to $[-\pi, \pi]$, so as similarity measure we use

$$d_\theta(\theta_1, \theta_2) = \sum_{i=1}^{m} 1 - \cos(\theta_1^i - \theta_2^i) \qquad (9)$$

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

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE



(a) (b) $d_\theta = 0.147$, $d_X = 0.053$, $d_I = 1.316$ (c) $d_\theta = 0.203$, $d_X = 0.052$, $d_I = 3.557$ (d) $d_\theta = 1.085$, $d_X = 0.063$, $d_I = 3.631$ (e) $d_\theta = 4.782$, $d_X = 0.060$, $d_I = 3.486$

Figure 3. Positive and negative paired examples. For each image in (b)–(e), the ± 1 label of the pair formed with (a) is based on the distance d_θ to the underlying parameters of (a), with similarity threshold $r = 0.25$. d_X and the root mean squared pixel distances d_I are given for reference.

found that this distance function, while not perfect, usually reflects our perception of pose similarity (see Figure 3 for examples).

After examining large numbers of images corresponding to poses with various distances, we set $r = 0.25$ and $\epsilon = 1$. An LSH query is therefore considered successful if it returns examples within $R = 0.5$ of the input. Analysis of the distribution of d_θ over pairs of training examples reveals that only about 0.05% of the pairs constitute a positive example by this criterion (the distribution appears to be roughly log-normal). Figure 3 shows 4 of 1,775,000 paired examples used to select hash functions; out of 11,270 features, 137 were selected using the thresholds $p_1 \geq .65$ and $p_2 \leq .38$. Based Based on Eq. 5, PSH was implemented via 150 hash tables using 18-bit hash functions.

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 1. Mean estimation error for synthetic test data, over 1000 examples. 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 the input distance, 1.659.

To quantitatively evaluate the algorithm's performance, we tested it on 1000 synthetic images, generated from the same model. Table 1 summarized the results with different methods of fitting a local model; 'linear' refers to a non-weighted linear model fit to the neighborhood. On average PSH searched 5100 candidates, about 3.4% of the data, per input example; in almost all cases, the true nearest neighbors under d_X were also the top PSH candidates.

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. Figure 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 PSH; 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 5), there is a dramatic improvement versus the estimate based on the single NN. The number of candidates examined by PSH was, as expected, significantly lower than for the synthetic images - about 2000, or 1.3% of the database. 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, 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 for evaluation. For most of the examples the pose estimate was accurate; on some examples it failed to various extents. Figures 4 and 5 show a number of examples, including two definite failures. Note that in some

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

Figure 4. Examples of upper body pose estimation (Section 4). Top row: input images. Middle row: top PSH match. 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.

Figure 5. 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 degrees of error in estimation.

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 **IEEE**

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.

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 regression, 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. [1, 13], 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 making previously infeasible example-based estimation paradigm 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. Moreover, it should be pointed out that there exist fast algorithms for approximate similarity search other than LSH. It remains an open question whether those algorithms can be modified for parameter sensitivity and become useful for estimation tasks in vision, replacing LSH in our framework.

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] V. Athitsos and S. Sclaroff. Estimating 3D Hand Pose from a Cluttered Image. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 432–439, Madison, WI, June 2003.

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

[3] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional space. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1000–1006, San Juan, PR, June 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] Curious Labs, Inc., Santa Cruz, CA. *Poser 5 - Reference Manual*, 2002.

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

[10] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimnensions: A texture classification example. In *International Conference on Computer Vision*, 2003. (to appear).

[11] 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.

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

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

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

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

[16] 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.

[17] 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.

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

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE