Cascaded Pose Regression

Piotr Dollár Peter Welinder Pietro Perona
California Institute of Technology
{pdollar,welinder,perona}@caltech.edu

Abstract

We present a fast and accurate algorithm for computing the 2D pose of objects in images called cascaded pose regression (CPR). CPR progressively refines a loosely specified initial guess, where each refinement is carried out by a different regressor. Each regressor performs simple image measurements that are dependent on the output of the previous regressors; the entire system is automatically learned from human annotated training examples. CPR is not restricted to rigid transformations: 'pose' is any parameterized variation of the object's appearance such as the degrees of freedom of deformable and articulated objects. We compare CPR against both standard regression techniques and human performance (computed from redundant human annotations). Experiments on three diverse datasets (mice, faces, fish) suggest CPR is fast (2-3ms per pose estimate), accurate (approaching human performance), and easy to train from small amounts of labeled data.

1. Introduction

Detection and localization are among the most useful functions of vision. Detection consists of giving a one-bit answer to the question "*Is object/category x in the image?*". Localization is a more subtle problem: in its simplest and most popular form [11], it consists of identifying the smallest rectangular region of the image that contains the object in question. This is perfectly sufficient for categories whose main geometric degrees of freedom in the image are translation and scale, such as frontal faces and pedestrians. More generally, one wishes to recover *pose*, that is a number of parameters that influence the image of the object. Most commonly pose refers to geometric transformations of rigid objects [23] including the configuration of articulated objects, for example the limbs of a human body [26, 14] or vehicle layout [21]. More broadly, pose is any set of systematic and parameterizable changes in the appearance of the object [5]. There are two distinct reasons for computing the pose of an object: (1) due to object variability, the only way to perform detection is to compute and factor out pose explicitly, (2) pose is the desired output of the vision module. In this work we are interested in the latter: we wish to estimate the pose of an object given its rough initial location, for example as provided by a tracker.

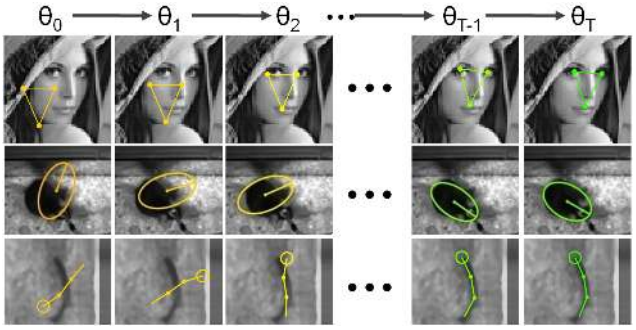

Figure 1. Object pose (green wire frame) is computed by cascaded pose regression (CPR) starting from a coarse initial guess (orange wire frame). The parameterization of pose is arbitrary and need only be consistent across training examples. CPR is implemented as a sequence of regressors progressively refining the estimate of the pose θ. At each step $t = 1 \dots T$ in the cascade, a regressor R^t computes a new pose estimate θ_t from the image and from the previous regressor's estimate θ_{t-1}. *Left:* Initial guess θ_0; *Right:* final estimate θ_T. Each row shows a test case culled from three different data sets. The same CPR code was trained to compute the pose of different objects/categories from a relatively small sample of hand-annotated training examples.

The predominant approach for object localization in position and scale is to use a 'sliding window', *i.e.*, repeating a binary classification task, "*Is object x at location y?*", for a fine-grained sampling of the pose parameters. Although this generates a large number of tests, sliding window methods can be made more efficient through cascades [28], distance transforms [13], branch-and-bound search [20] and coarse to fine approaches [15]. Such methods can can be extended to more complex notions of pose by repeatedly answering queries of the form "*Is object x at location y with pose θ?*", one for each partition of the pose θ. For example, for face detection it is common to train a separate classifier for different levels of out of plane rotation [28]. Of course this leads to a combinatorial explosion of tasks, and although efficient search strategies can help [16], ultimately such ap-

978-1-4244-6985-7/10/$26.00 ©2010 IEEE

proaches may not scale to more complex notions of pose.

In this work, given a rough estimate of the object location, we directly answer the question "*What is the pose θ of object x?*", recovering the pose without performing a potentially expensive and branching search. In principle, standard regression techniques do exactly this [17, 10]. Although for certain taks in computer vision regression has been successful [30, 1], its applicability to more general pose estimation remains unclear. As in boosted regression [17, 10, 30], we propose to learn a fixed linear sequence (cascade) of weak regressors (random ferns in our case). The key difference from previous iterative regression approaches is the use of *pose-indexed* features [16]: features whose output depends on both the image data and the current estimate of pose. By assuming certain *weak invariance* properties of the pose-indexed features, we derive a principled algorithm for pose estimation we call cascaded pose regression (CPR). We prove CPR converges at an exponential rate under a much weaker notion of weak learnability than is typically required for boosted regression. Accurate models can be learned with surprisingly small amounts of data ($O(100)$ labeled training examples). CPR is fast (2-3ms per pose estimate), accurate (approaching human performance), and easily trained on diverse object categories.

Our main contributions are: (1) A fast cascaded pose regression algorithm that produces accurate pose estimates on a wide variety of object categories, described in detail in Sec. 2. (2) Using redundantly annotated data we evaluate the performance of human annotators and define a perceptual distance function to compare pose annotations, described in Sec. 3. (3) A comprehensive experimental evaluation of the algorithm and promising results on a number of datasets in Sec. 4. We begin with related work below.

1.1. Related Work

The use of features that change as scene or object information is gathered has a long history in computer vision. Dickmanns and Graefe [9] proposed a real-time control scheme that computed features sparsely at locations likely to contain useful information for a dynamic vision system. Goncalves *et al*. [18] computed 3D arm pose iteratively, refining 2D image features with each improved pose estimate; more recently, Ramanan [26] used similar ideas in a multi-stage pose estimation procedure. Fleuret and Geman [16] coined the term 'pose-indexed features' to refer to features defined relative to a given pose; in this work we adopt the same terminology. However, unlike [16] we use pose-indexed features for regression as opposed to classification, allowing us to perform pose estimation directly.

Early work in pose estimation includes snakes [19], template matching [29], and active appearance models [8]. Although extensions similar in spirit to CPR that use learning to drive the optimization have been proposed [27], typically these methods require manually defined energy functions that measure goodness of fit. Another relevant approach is structured output prediction [4], which provides a principled formulation for *learning* to answer "*Is object x at location y with pose θ?*". However, as with standard classification efficient search/inference techniques are still required [20, 4]. Many modern detection approaches involve decomposing objects into component parts, detecting the parts separately and then combining them by means of a flexible parts model [7, 12, 5]. Such approaches, although effective at detecting articulated objects, have not been shown to return accurate pose estimates. In recent work, Ali *et al*. [2] used pose-indexed features to perform detection of articulated objects, integrating such features directly into a boosted cascade, and Özuysal *et al*. [25] used pose estimation *prior* to detection, conditioning their classifier on the pose estimate. Both approaches present new and interesting directions for integrating pose into detection; however, in this work we focus on the pose estimation problem itself.

2. Cascaded Pose Regression

In order to clearly discuss object pose and appearance, we assume there exists some unknown image formation model $G : \mathcal{O} \times \Theta \rightarrow \mathcal{I}$ that takes an object appearance $o \in \mathcal{O}$ and pose $\theta \in \Theta$, and generates an image $I \in \mathcal{I}$. We never have explicit access to G or o; however, they are necessary for the derivations that follow. For example, we can write $I_{\theta_1} = G(o, \theta_1)$ and $I_{\theta_2} = G(o, \theta_2)$ to denote two images of the same object o in two configurations θ_1 and θ_2. We assume that $G(o_1, \theta_1) = G(o_2, \theta_2)$ iff $o_1 = o_2$ and $\theta_1 = \theta_2$, otherwise uniquely estimating pose may not be possible.

We require that Θ along with the operation $\circ$ form a group. Given two poses θ_1, θ_2, we write $\theta = \theta_1 \circ \theta_2$ to denote a novel pose formed by combining θ_1 and θ_2, $\overline{\theta}$ to denote the inverse of θ, and e to denote the identity element. To measure relative error between two poses, we require a function $d : \Theta \times \Theta \rightarrow \mathbb{R}$ where $d(\theta_1, \theta_2)$ can depend only on the relative pose $\overline{\theta}_1 \circ \theta_2$, or equivalently that $d(\theta_\delta \circ \theta_1, \theta_\delta \circ \theta_2) = d(\theta_1, \theta_2)$ for all $\theta_1, \theta_2, \theta_\delta \in \Theta$.

2.1. Pose-Indexed Features and Weak Invariance

As mentioned, throughout this work we rely on pose-indexed features. A pose-indexed features is simply a function of the form $h : \Theta \times \mathcal{I} \rightarrow \mathbb{R}$. We say h is *weakly invariant* if $\forall \theta, \theta_\delta \in \Theta$ the following holds:

$$h(\theta, G(o, e)) = h(\theta_\delta \circ \theta, G(o, \theta_\delta)), \tag{1}$$

or equivalently, h is weakly invariant if $\forall \theta_1, \theta_2, \theta_\delta \in \Theta$:

$$h(\theta_1, G(o, \theta_2)) = h(\theta_\delta \circ \theta_1, G(o, \theta_\delta \circ \theta_2)). \tag{2}$$

It is easy to show that (2) holds if and only if (1) holds. Another way of stating the above is that $h(\theta_1, G(o, \theta_2))$ de-

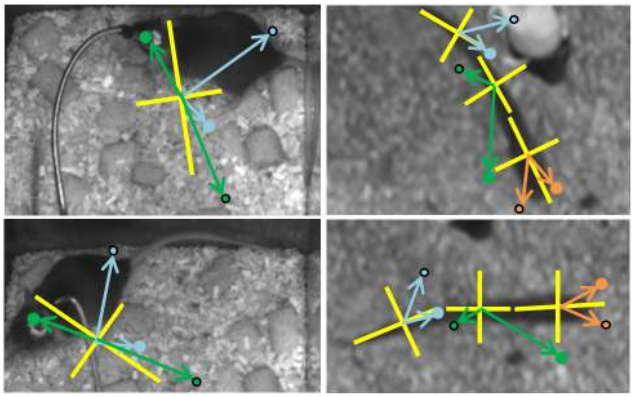

Figure 2. Pose-indexed features. *Left:* Mice described by a 1-part pose model. *Right:* 3-part pose model of zebra fish. The yellow crosses represent the coordinate system defined by the current estimate of the pose of the object (which does not have to be centered on the object). The colored arrows show control points defined relative to the pose coordinates. The weakly pose-invariant features used in this paper were defined as the difference in pixel values at two control points relative to the pose.

pends only on the object o and the relative pose $\overline{\theta}_1 \circ \theta_2$ between the input pose θ_1 and true pose θ_2. In other words, h is weakly invariant if its output is constant given a consistent (not necessarily correct) estimate of the pose. Composing weakly invariant featurs using standard operations results in a features that are themselves weakly invariant.

Note that invariance as defined above is a much weaker requirement than general pose invariance, which could be stated as follows: $h(G(o,\theta)) = h(G(o,\theta_\delta \circ \theta))$. Designing invariant function that satisfy the latter definition is exceedingly difficult, while our definition requires invariance only when given a consistent estimate of the pose. It is also worth comparing (2) to the 'stationarity assumption' introduced in [16]. Using the notation defined here, a non-probabilistic form of the stationarity assumption can be written as $h(\theta_1, G(o,\theta_1)) = h(\theta_2, G(o,\theta_2))$. In other words it states $h(\theta, G(o,\theta))$ is constant regardless of the value of θ. Observe that (2) is a very natural, albeit stronger, generalization of the stationarity assumption.

Our weak invariance assumption justifies the derivations that follow and allows us to prove strong convergence rates for the resulting algorithm. Under ideal conditions, we can prove (2) holds; however, as in [16], we observe that in practice (2) will frequently be violated. Nevertheless, as we shall demonstrate, the algorithm derived using the weak invariance assumption is very effective in practice.

Pose-Indexed Control Point Features In all our experiments we use the extremely simple and fast to compute control point features [22, 24]. In our implementation, each control point feature is computed as the difference of two image pixels at predefined image locations. More specifically, each feature h_{p_1,p_2} is defined by two image locations p_1 and p_2 and is evaluated by computing $h_{p_1,p_2}(I) = I(p_1) - I(p_2)$, where $I(p)$ denotes the grayscale value of image I at location p.

Input: Image I, initial pose θ^0
1: **for** $t = 1$ to T **do**
2: $x = h^t(\theta^{t-1}, I)$ // compute features
3: $\theta_\delta = R^t(x)$ // evaluate regressor
4: $\theta^t = \theta^{t-1} \circ \theta_\delta$ // update θ^t
5: **end for**
6: Output θ^T

Figure 3. Evaluation of Cascaded Pose Regression.

Aside from their speed and surprising effectiveness in real applications [24], the advantage of the above features is they are straightforward to index by pose. For example, suppose object pose is specified by a translation, rotation, scale and aspect ratio (or some subset of these parameters). For each pose θ, we can define an associated 3×3 homography matrix H_θ, express p in homogeneous coordinates, and define $h_{p_1,p_2}(\theta, I) = I(H_\theta p_1) - I(H_\theta p_2)$. We can easily extend this approach to articulated objects where each part has a rotation, scale and aspect ratio by associating a separate homography matrix with each part. See Figure 2.

In the appendix (available on the project website) we show that $h(\theta, I)$ defined in the manner above is weakly invariant under certain assumptions. In general, however, designing weakly invariant pose-indexed features can be quite challenging and requires careful consideration when applying our proposed framework to novel problems.

2.2. Cascaded Pose Regression

We now describe the evaluation and training procedures for a cascaded pose regressor $R = (R^1, \ldots, R^T)$, shown in Figures 3 and 4, respectively. We will train a cascaded regressor $R = (R^1, \ldots, R^T)$, such that, given an input pose θ^0, $R(\theta^0, I)$ is evaluated by computing:

$$\theta^t = \theta^{t-1} \circ R^t(h^t(\theta^{t-1}, I)), \tag{3}$$

from $t = 1 \ldots T$ and finally outputting θ^T (see Figure 3). Each component regressor R^t is trained to attempt to minimize the difference between the true pose and the pose computed by the previous components using (pose-indexed) features h^t. Our goal is to optimize the following loss:

$$\mathcal{L} = \sum_{i=1}^{N} d(\theta_i^T, \theta_i). \tag{4}$$

We begin by computing $\theta^0 = \arg\min_\theta \sum_i d(\theta, \theta_i)$, and set $\theta_i^0 = \theta^0$ for each i. θ^0 is the single pose estimate that gives the lowest training error without relying on any component regressors. We now describe the procedure for training R^t

Input: Data (I_i, θ_i) for $i = 1 \dots N$
$\theta^0 = \arg\min_\theta \sum_i d(\theta, \theta_i)$
$\theta_i^0 = \theta^0$ for $i = 1 \dots N$
for $t = 1$ to T **do**
  $x_i = h^t(\theta^{t-1}, I_i)$
  $\widetilde{\theta}_i = \overline{\theta}_i^{t-1} \circ \theta_i$
  $R^t = \arg\min_R \sum_i d(R(x_i), \widetilde{\theta}_i)$
  $\theta_i^t = \theta_i^{t-1} \circ R^t(x_i)$
  $\epsilon_t = \sum_i d(\theta_i^t, \theta_i) / \sum_i d(\theta_i^{t-1}, \theta_i)$
  If $\epsilon_t \geq 1$ stop
end for
Output $R = (R^1, \dots, R^T)$

Figure 4. Training for Cascaded Pose Regression.

given $R^1, \dots, R^{t-1}$. In each phase t, we begin training by randomly generating the pose-indexed features h^t and computing $x_i = h^t(\theta_i^{t-1}, I_i)$ for each training example I_i with the previous pose estimate θ_i^{t-1}. Our goal is to learn a regressor R^t such that $\theta_i^t = \theta_i^{t-1} \circ R^t(x_i)$ minimizes the loss in (4). After some manipulation, we can write this as:

$$R^t = \arg\min_R \sum_i d(R(x_i), \widetilde{\theta}_i), \tag{5}$$

where $\widetilde{\theta}_i = \overline{\theta}_i^{t-1} \circ \theta_i$. We can solve for R^t using standard regression techniques, moreover, since R^t needs to only slightly reduce the error, we can train separate single-variate regressors for each coordinate of $\widetilde{\theta}$ and simply keep the best one. In this work we rely on random regression ferns, described below.

After training R^t, we apply (3) to compute θ_i^t for use in the next phase of training. If the regressor R^t was unable to reduce the error training stops. Let:

$$\epsilon_t = \sum_i d(\theta_i^t, \theta_i) / \sum_i d(\theta_i^{t-1}, \theta_i). \tag{6}$$

If $\epsilon_t \geq 1$ training stops, otherwise we can continue training for T phases or until the error drops below a certain target value. The full training procedure is given in Figure 4.

Random Fern Regressors Encouraged by the success of random ferns for classification [24] and random forests for regression [6], we train a random fern regressor at each stage in the cascade. A fern regressor takes an input vector $x_i \in \mathbb{R}^F$ and produces an output $y_i \in \mathbb{R}$. It is created by randomly picking S elements from the F-dimensonal feature vector with replacement, and then sampling S thresholds randomly. The jth element of x_i is compared to the jth threshold to create a binary signature of length S. Thus, each x_i ends up in one of 2^S bins. The y prediction for a bin is the mean of the y_i's of the training examples that fall into the bin. At each stage in the cascade, the best fern in terms of training error is picked from a pool of R randomly generated ferns.

Pose Clustering Depending on the initialization of the pose before applying CPR, the algorithm sometimes fails to estimate the correct pose. However, more often than not, just re-running CPR with a different initial pose yields a reasonable estimate. Thus, we used a simple "pose clustering" heuristic to improve the performance of the algorithm. For each image, CPR was run K times with different random initial poses. Then, after all K runs, the pose in the highest density region of pose-space was picked as the final prediction of the algorithm. We used a simple Parzen window approach with a Gaussian kernel of width 1 (using noramlized distances described in Section 3) to estimate the density at each pose as compared to the other $K - 1$ poses.

Convergence Rate We prove that our iterative scheme will converge under fairly weak assumptions, and furthermore, show that the rate of convergence is exponential in the weak errors of the component regressors. The proof is similar in spirit to the proofs for convergence of boosted regressors [17, 10], but requires a weaker notion of weak learnability. Here we highlight the main findings (see appendix on project webpage for full proof).

Let h represent a set of standard (not pose-indexed) features. We define the relative error of a regressor R on a data set (I_i, θ_i) as $\epsilon = \sum_i d(R(h(I_i)), \theta_i) / \sum_i d(\theta, \theta_i)$, for the θ which minimizes the denominator. Thus, if R performs better then returning the single uniform prediction θ, $\epsilon < 1$. Fairly straightforward convergence proofs for both CPR and boosted regression [17] require that we have access to a weak learner, that, given a data set (I_i, θ_i) can output a regressor with relative error $\epsilon \leq \beta$ for some $\beta < 1$. Under these conditions, the rate of convergence for both CPR and boosted regression is given by $\epsilon^T \leq \beta^T$.

The primary difference between the convergence rates of CPR and boosting regression lies in the strength of the weak learnability assumption. Let $I_i = G(o_i, \theta_i')$ for some unknown o_i. In CPR, we need access to a weak learner that can output a regressor with $\epsilon \leq \beta$ on a training set (I_i, θ_i) *only* if $\theta_i = \theta_i'$. For boosted regression, we need access to a weak learner that can output a regressor with $\epsilon \leq \beta$ on *arbitrary* training sets (I_i, θ_i) where θ_i need not equal θ_i'. Thus, although both CPR and boosted regression converge exponentially at some rate β^T as long as the weak learnability assumption is satisfied, in practice the base of the exponent β is much lower for CPR.

2.3. Data Augmentation

Utilizing pose indexed features, we can artificially simulate a large amount of data from the N training samples by simply using different initial estimates for the pose. This allows us to avoid the combinatorial explosion of data that would be required if we needed to observe every object in every pose. Suppose we are given training samples (I_i, θ_i) for

$i = 1 \ldots N$ and wish to simulate additional data. Recall that each image I_i is generated using $G(o_i, \theta_i)$ where o_i is the unknown object appearance. Using the training data, we can estimate the distribution $\mathcal{D}$ of the poses θ_i (or use their empirical distribution); we would like to use $\mathcal{D}$ to augment our training set by sampling $\theta_j \sim \mathcal{D}$ and generating new training images $I_{ij} = G(o_i, \theta_j)$. Although we do not have access to either G or o_i, we can actually achieve an identical effect by taking advantage of our pose-indexed features being weakly invariant. We formalize this below.

When training with the un-augmented data we optimize the following loss: $\mathcal{L}(R) = \sum_{i=1}^{N} d(R(h(\theta, I_i)), \theta_i)$ where θ is the initial pose. Suppose we could explicitly generate additional images using G by sampling $\theta_j \sim \mathcal{D}$ and generating novel images $I_{ij} = G(o_i, \theta_j)$. In effect, we could optimize

$$\mathcal{L}(R) = \sum_i E_{\mathcal{D}}\big[d(R(h(\theta, G(o_i, \theta_j))), \theta_j)\big], \qquad (7)$$

where $E_{\mathcal{D}}$ denotes expectation over $\mathcal{D}$. Of course, in practice we do not have access to G. However, we can achieve an identical effect without needing to explicitly compute G. Below we prove that $\overline{\theta}_j \circ R(\theta, G(o, \theta_j)) = \overline{\theta}_i \circ R(\theta', G(o, \theta_i))$, where $\theta' = \theta_i \circ \overline{\theta}_j \circ \theta$. Plugging into (7) and re-arranging gives:

$$\mathcal{L}(R) = \sum_i E_{\mathcal{D}}\big[d(R(h(\theta_i \circ \overline{\theta_j} \circ \theta, I_i)), \theta_i)\big]. \qquad (8)$$

In other words, training R with the original I_i but with an initial pose estimate $\theta' = \theta_i \circ \overline{\theta}_j \circ \theta$ is exactly equivalent to training with explicitly generated novel images I_{ij}. In practice, we approximate the loss in (8) by sampling a finite set of initial poses for each training example.

To complete the above derivation, we prove that $\forall \theta_1, \theta_2, \theta_1^0, \theta_2^0$, if $\overline{\theta}_1 \circ \theta_1^0 = \overline{\theta}_2 \circ \theta_2^0$, then the following holds:

$$\overline{\theta}_1 \circ R(h(\theta_1^0, G(o, \theta_1))) = \overline{\theta}_2 \circ R(h(\theta_2^0, G(o, \theta_2))) \qquad (9)$$

We prove by induction that $\overline{\theta}_1 \circ \theta_1^t = \overline{\theta}_2 \circ \theta_2^t$ for every t, where θ_1^t and θ_2^t are defined as in (3). The base case ($t = 0$) is true by definition. We now show that if the above holds for $t - 1 > 0$, it also holds for t. Proof:

$$\overline{\theta}_1 \circ \theta_1^t = \overline{\theta}_1 \circ \theta_1^{t-1} \circ R^t(h(\theta_1^{t-1}, G(o, \theta_1))) \qquad (10)$$

$$= \overline{\theta}_1 \circ \theta_1^{t-1} \circ R^t(h(\overline{\theta}_1 \circ \theta_1^{t-1}, G(o, e))) \qquad (11)$$

$$\overline{\theta}_2 \circ \theta_2^t = \overline{\theta}_2 \circ \theta_2^{t-1} \circ R^t(h(\theta_2^{t-1}, G(o, \theta_2))) \qquad (12)$$

$$= \overline{\theta}_2 \circ \theta_2^{t-1} \circ R^t(h(\overline{\theta}_2 \circ \theta_2^{t-1}, G(o, e))) \qquad (13)$$

Therefore, if $\overline{\theta}_1 \circ \theta_1^{t-1} = \overline{\theta}_2 \circ \theta_2^{t-1}$ then $\overline{\theta}_1 \circ \theta_1^t = \overline{\theta}_2 \circ \theta_2^t$, thus completing the proof.

3. Human Annotations and Ground Truth

We obtained three pose-labeled datasets: Mice, Fish and Faces. The *Mice* dataset consisted of 3000 black mice labeled in top-view images (with 1-3 mice per image). Pose

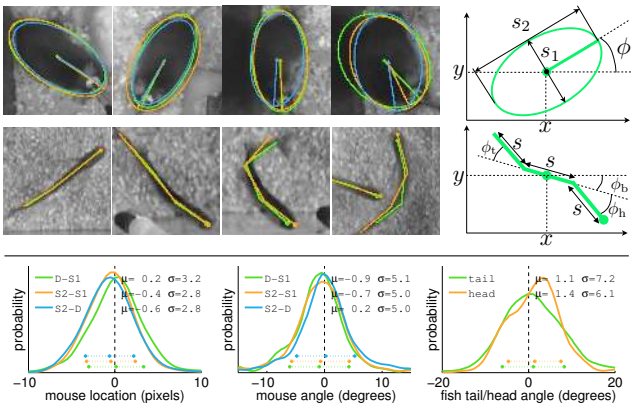

Figure 5. Pose labels provided by human annotators. *Top-left*: Annotations for the Mice and Fish datasets provided by different annotators (color denotes annotator). *Top-right*: Parameterization of the poses. The mouse pose is an ellipse at location (x, y) with orientation ϕ, scale s_1 and aspect ratio s_1/s_2. The fish pose is a 3-part model where the body (middle) part is centered at location (x, y) with orientation ϕ_b, and the tail and head parts have angles ϕ_t and ϕ_h respectively w.r.t. to the body part. The scale s is the length of the parts. *Bottom*: Distributions of differences in human-provided pose labels for the location and orientation of the mice (three annotators: S1, S2, D), and the tail and head angles for the fish (two annotators). The estimated mean and standard deviation of each distribution is denoted by μ and σ respectively.

was specified by the location, orientation, scale, and aspect ratio of an ellipse fitted around each mouse, see Fig. 5 (top). The *Fish* dataset consisted of 38 top-view images of zebrafish swimming in an aquarium, with 5 fish per image. With reflection, this gives a total of 380 examples. The pose of each fish was annotated by fitting a 3-part model specified by the location, orientation and scale of a central body part, and the angles of the tail and head with respect to the body, see Fig. 5 (middle). For the *Faces* dataset we used Caltech 10,000 Web Faces [3], where each face was labeled by four points (eyes, mouth, nose). We used the first three coordinates (the nose was somewhat inconsistent) to define a pose with the same parameterization as an ellipse.

In addition to training and evaluating CPR, we used *redundantly* annotated images for measuring human performance and defining a perceptually meaningful distance measure between poses. Three annotators provided redundant labels for 750 mice, and two annotators provided labels for all the fish, see Fig. 5 (top) for some examples. To quantify annotator consistency, for each pose parameter we computed the distributions of pose-parameter differences, as shown in Fig. 5 (bottom). Observe that the annotations are consistent and unbiased ($\mu \approx 0$) and the differences in individual pose parameters are normally distributed. The latter motivates the use of the standard deviation as normalization in a perceptually meaningful distance measure.

In order to weigh errors in estimating the differ-

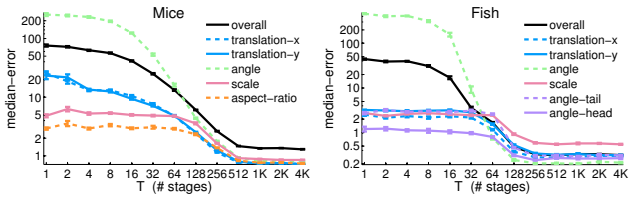

Figure 6. Performance *vs*. the number of phases T in the regression cascade. Overall median error plotted in black; error of individual pose parameters plotted in color. *Mice*: Initially only the translation parameters are refined, next CPR begins predicting orientation, finally after $\sim$128 iterations CPR also begins refining scale and aspect ratio. *Fish*: Due to the elongated 3-part pose model, CPR determines the orientation of the model before concentrating on the position, scale and part angles. No overfitting is observed with increasing T.

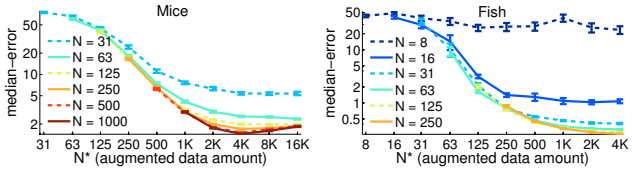

Figure 7. Effect of data augmentation (Sec. 2.3). Each curve shows the effect of augmenting N actual training examples to different sizes $N^* \geq N$. *Mice*: Performance is reasonable with just $N \geq 125$ training examples augmented to $N^* = 4000$ total examples (and maximized as soon as $N \geq 250$). *Fish*: Performance is maximized when $N \geq 64$ training examples are used (again augmented to $N^* = 4000$ total examples); in fact, since the training images were mirrored, this corresponds to 32 actual annotations. Thus, although the total amount of augmented data CPR requires is massive, the amount of actual annotated data needed is very small.

ent pose parameters equally against each other, we define the distance between two poses as $d(\theta_1, \theta_2) = \sqrt{\frac{1}{D}\sum_{i=1}^{D}\frac{1}{\sigma_i^2}\left(\theta_1^i - \theta_2^i\right)^2}$, where D is the number of pose parameters. Here σ_i^2 denotes the variance of the differences between annotations of the ith pose parameter, estimated from the difference distributions shown in Fig. 5. These weights were used both for evaluation and for training CPR. The above normalization assumes that the pose parameters are uncorrelated. The expected squared distance between two human annotators is 1. We defined a pose estimate with a distance from the ground truth greater than 2.5 to be a *failure*, as we observed that with 99% probability two human annotations of the same object were within a normalized distance of 2.5.

4. Experiments

We performed experiments on the three datasets described above: Mice, Faces and Fish. We divided the Mice and Faces datasets into training, validation and test sets of 1000 images each. We divided the Fish dataset into a training set

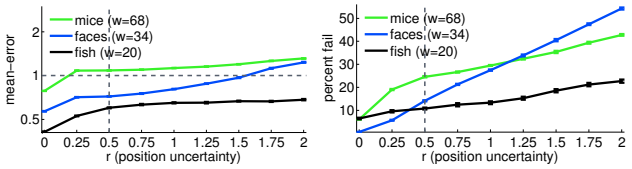

Figure 8. Performance as a function of the uncertainty in the initial position r of a (simulated) detector (see text for details). *Left*: Mean error as a function or r increases gradually, except for the faces where more textured backgrounds likely make pose estimation more challenging with larger offsets. *Right*: The failure rate increases smoothly but rapidly as detector uncertainty increases.

of 250 images and a test set of 130 images (but no validation set). In addition, we applied an 8×8 median filter to the fish images to remove salt and pepper noise from the background. All reported experiments were averaged over 25 trials, each performed with a different random seed.

We measured the error of a single annotation using the perceptual distances defined in Section 3. If the error is above 2.5 we define the pose estimate to be a *failure*. We report overall error in terms of the *percent failure rate* and the *mean-error* of the remaining examples. Alternatively, we report *median-error* when we wish to describe performance using a single curve.

The main parameters of CPR are the amount of training data N, the total amount of data after data augmentation N^*, the number of phases in the cascade T, the fern depth S, the number of ferns R and features F generated at each stage of training, and the number of restarts K. Using the Mice and Faces validation sets we found $(N^* = 4000, T = 512, S = 5, R = 64, F = 64, K = 16)$ to be a good tradeoff between performance and speed on both datasets. All subsequent experiments, on all datasets, used the above parameter settings unless otherwise noted.

Cascade depth: The influence of the number of phases T on the error is shown in Fig. 6. The algorithm converges after 512 stages for all datasets, including the faces (not shown), and does not overfit. The lack of overfitting is in stark contrast to standard boosted regression algorithms which tend to strongly overfit and require careful tuning of a "learning rate" parameter [17].

Data-augmentation: Fig. 7 shows the advantage of the data augmentation scheme described in Section 2.3. With only 250 training examples it is possible to match the performance achieved using all 1000 training examples on the Mice dataset. On the Fish dataset the benefit of data augmentation was even more striking: just 32 pose labels were sufficient to achieve human performance.

Translational uncertainty: We expect a detector/tracker to provide a center position estimate x to CPR. We assume the detector always returns an estimate x that is within a maximum distance rw of the true object position x^*, *i.e.*, $||x - x^*||_2 \leq rw$, where w denotes object width

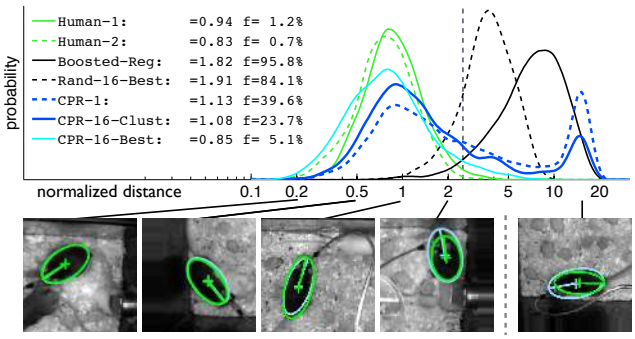

Figure 9. **Mice dataset**: Human agreement on this dataset is high, with a mean error $\mu < 1$ and a failure rate $f \approx 1\%$. *Boosted-Reg* fails on almost all test examples. Choosing the best of 16 random poses (*Rand-16-Best*) results in performance slightly better than *Boosted-Reg*, but much worse than CPR. The distribution of errors of *CPR-1* is bimodal with 39.6% failures; however, the mean (excluding failures) is $\mu = 1.13$, which is not far off from human performance. Running CPR with clustering (*CPR-16-Clust*) reduces failures substantially. Having an oracle pick the best of 16 pose predictions (*CPR-16-Best*) removes most remaining failures, a property that can be exploited in tracking systems where dynamic information is available. The images in the bottom row show examples of pose estimates (green ellipse) from *CPR-16-Clust* at different distances from the ground truth (blue ellipse). We observed that many of the failure cases were caused by orientation errors of 180°, as in the right-most image.

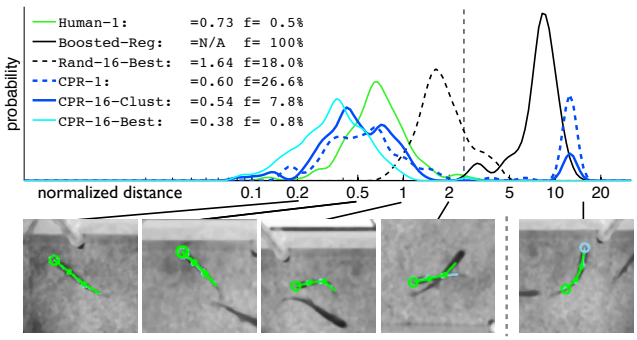

Figure 10. **Fish dataset**: Results are qualitatively similar to the mice data (see Fig. 9). Failures are again primarily 180° orientation errors (which can be removed by clustering or by incorporating dynamic information). Interestingly, except for the few failure cases, CPR outperforms the human annotator that redundantly labeled the same data after being given identical labeling instructions. Neither set of annotations looks sloppy, rather there apparently must be some slight bias between the annotators, whereas the algorithm learns to mimic the first annotator quite closely.

measured at the root part and r defines the uncertainty of the detector. We simulate such a detector by sampling initial estimates x_i uniformly from a circular region of radius rw centered on x_i^* for each example i. Results are shown in Fig. 8. Mean performance (excluding failures) degrades gradually as the uncertainty r increases; however, the failure rate increases more quickly. Throughout all experiments we use a simulated detector with $r = .5$ as we expect most detection systems to return a position estimate that is within half an object width of the true object position.

Performance: Results on the fish, mice and faces are shown in Figures 9–11. We plot the full distribution of errors, and list both the mean errors μ (excluding the failures) and the failure rates f. For each dataset, we compare the following approaches:

Human: human versus human performance.
Boosted-Reg: boosted regression [17] using same features as CPR.
Rand-16-Best: oracle selects the best of 16 random poses.
CPR-1: CPR with a single ($K = 1$) starting pose.
CPR-16-Clust: CPR with 16 starting poses followed by clustering.
CPR-16-Best: CPR with 16 starting poses, oracle selects best.

CPR-16-Clust is the basic approach described in this paper while *CPR-16-Best* shows performance when an outside source of information is available to select the best of 16 poses computed by CPR (*e.g.*, temporal consistency information). Overall, the performance of *CPR-16-Clust* is close to that of human annotators, except for a 5%-25% failure rate depending on the dataset. However, many failure cases in the Mice and Fish datasets are due to orientation errors of 180°, a problem that can be alleviated by dynamic constraints from a tracking system. Indeed, *CPR-16-Best* tends to have a very low failure rate of 1%-5%. For comparison, *Boosted-Reg* and *Rand-16-Best* perform very poorly.

Implementation: Using a Matlab implementation of CPR, it takes about 3 minutes to train the entire system on $N = 1000$ image/pose pairs using the default parameters. Testing is also very fast, averaging 2-3 ms per image with default parameters and $K = 1$ starting poses. We will post all code (which is fairly small) on the project website.

5. Discussion and Conclusions

We presented a new algorithm, cascaded pose regression (CPR), to compute the 2D pose of an object from a rough initial estimate. The key to CPR's success appears to be the use of pose-indexed features whose values depends on the current estimate of the pose. Training CPR takes only a few minutes and computing pose a few milliseconds on a standard machine. CPR is insensitive to exact parameter setting; indeed, identical parameters were used for all three datasets. Moreover, CPR can learn effective models with very little training data (~ 100 training samples).

Experiments carried out on three datasets (faces, mice, fish) with different object and background statistics demonstrated that CPR can learn diverse models of pose with a median error comparable to that of a skilled human annotator. While this is very encouraging, we will need to experiment with more categories before we can claim general applicability. The failure rate is higher than that of human

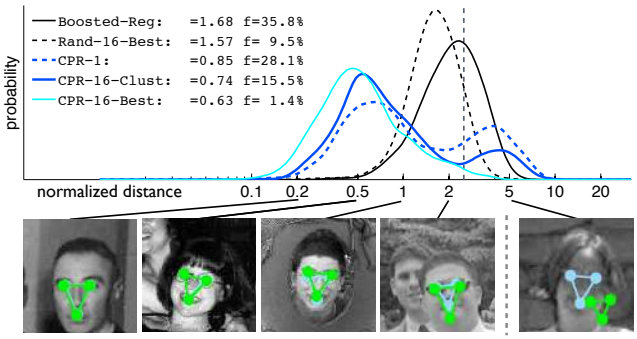

Figure 11. **Faces dataset**: Results are qualitatively similar to the Mice and Fish datasets (see Fig. 9 and Fig. 10). Since only one set of labels was available, no human performance curves are shown and we had to manually set the distance weights.

annotators; however, failures can be alleviated by clustering pose estimates or through use of external information such as temporal constraints.

Future work includes incorporating CPR into detection or tracking systems. Our experiments suggest that a detector which is able to initialize CPR within a patch about the same size as the object is sufficient for CPR to produce good estimates of pose. Merging CPR into a tracking by detection system could result in a robust new approach for tracking of articulated objects.

Acknowledgments

We thank Dayu Lin and David Anderson for the mice data, support and motivation; and Konstantin Startchev and Jacob Engelmann for providing the fish data and helpful discussions. This work was supported by ONR MURI Grant #N00014-06-1-0734 and ONR/Evolution Grant #N00173-09-C-4005. PD was supported by a Caltech Endowment Award #101240.

References

[1] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *CVPR*, 2004. 2

[2] K. Ali, F. Fleuret, D. Hasler, and P. Fua. Joint pose estimator and feature learning for object detection. In *ICCV*, 2009. 2

[3] A. Angelova, Y. Abu-Mostafa, and P. Perona. Pruning training sets for learning of object categories. In *CVPR*, 2005. 5

[4] M. Blaschko and C. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008. 2

[5] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose ann. In *ICCV*, 2009. 1, 2

[6] L. Breiman. Random forests. *Machine learning*, 2001. 4

[7] M. Burl, M. Weber, and P. Perona. A prob. approach to object recog. using local photometry and global geometry. In *ECCV*, 1998. 2

[8] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *PAMI*, 23(6), 2001. 2

[9] E. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1(4):223–240, 1988. 2

[10] N. Duffy and D. P. Helmbold. Boosting methods for regression. *Machine Learning*, 47(2-3):153–200, 2002. 2, 4

[11] M. Everingham et al. The 2005 pascal visual object classes challenge. In *First PASCAL Machine Learning Challenges Workshop, MLCW*, pages 117–176, 2005. 1

[12] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detect. with discr. trained part-based models. *PAMI*, 2010. 2

[13] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *CVPR*, 2000. 1

[14] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progr. search space reduction for human pose est. In *CVPR*, 2008. 1

[15] F. Fleuret and D. Geman. Coarse-to-fine face detection. *IJCV*, 41(1-2):85–107, 2001. 1

[16] F. Fleuret and D. Geman. Stationary features and cat detection. *JMLR*, 9:2549–2578, 2008. 1, 2, 3

[17] J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001. 2, 4, 6, 7

[18] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3d. In *ICCV*, pages 764–770, 1995. 2

[19] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988. 2

[20] C. Lampert, M. Blaschko, T. Hofmann, and S. Zurich. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008. 1, 2

[21] Y. Li, L. Gu, and T. Kanade. A robust shape model for multi-view car alignment. In *CVPR*, 2009. 1

[22] F. Moutarde, B. Stanciulescu, and A. Breheret. Real-time visual detection of vehicles and pedestrians with new efficient AdaBoost features. In *IROS*, volume 26, 2008. 3

[23] R. Nevatia and T. O. Binford. Description and recognition of curved objects. *Artif. Intell.*, 8(1):77–98, 1977. 1

[24] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *PAMI*, 2009. 3, 4

[25] M. Özuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009. 2

[26] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*, 2006. 1, 2

[27] J. Saragih and R. Goecke. A nonlinear discriminative approach to aam fitting. In *ICCV*, 2007. 2

[28] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004. 1

[29] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *IJCV*, 8(2):99–111, 1992. 2

[30] S. Zhou, B. Georgescu, X. Zhou, and D. Comaniciu. Image based regression using boosting method. In *ICCV*, 2005. 2