Steerable Part Models

Hamed Pirsiavash Deva Ramanan
Department of Computer Science, University of California, Irvine
{hpirsiav,dramanan}@ics.uci.edu

Abstract

We describe a method for learning steerable deformable part models. Our models exploit the fact that part templates can be written as linear filter banks. We demonstrate that one can enforce steerability and separability during learning by applying rank constraints. These constraints are enforced with a coordinate descent learning algorithm, where each step can be solved with an off-the-shelf structured SVM solver. The resulting models are orders of magnitude smaller than their counterparts, greatly simplifying learning and reducing run-time computation. Limiting the degrees of freedom also reduces overfitting, which is useful for learning large part vocabularies from limited training data. We learn steerable variants of several state-of-the-art models for object detection, human pose estimation, and facial landmark estimation. Our steerable models are smaller, faster, and often improve performance.

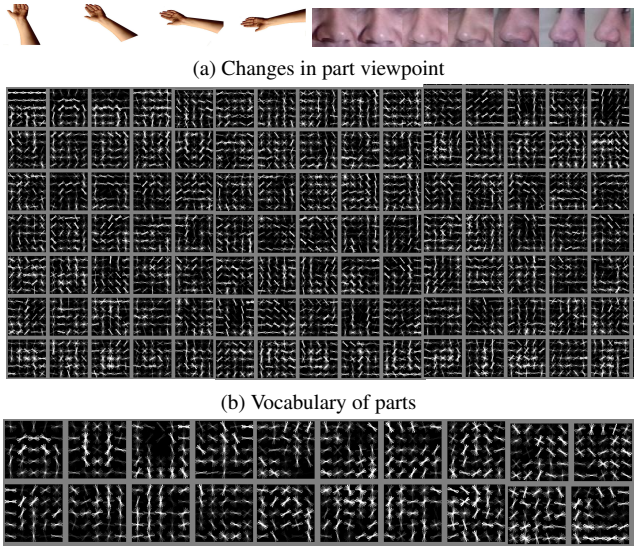
1. Introduction

Part-based models provide a promising framework for capturing variation in the appearance of an object. They do so by reasoning about local appearance using part templates; by composing different parts together in shifted positions, one can model a variety of global appearances. However, one will likely need a large set of parts to model changes in view point, deformation, scale, etc., across thousands of object categories. One challenge that lies ahead is the development of scalable representations that efficiently capture such large-scale part vocabularies.

Most current part models are implemented as templates defined on gradient features such as HOG [3]. Often these templates are trained using linear classifiers (such as an SVM), resulting in very high-dimensional learning problems. We argue that by conceptualizing these problems as one of learning *spatial filters* rather than high-dimensional parameter vectors, one can leverage the considerable body of work in image processing for developing efficient representations [8, 14, 11].

Typical approaches for reducing the size of a part vocabulary include vector quantization (e.g., visual words [18]). We show that one can also use linear subspace methods as an alternative form of compression. We represent a large vocabulary of parts as linear combinations of a small set of basis filters. One can use a small number of basis filters to "steer" across a variety of factors, including viewpoint, scales, and even semantic part types. We show that one can encode steerable and separable constraints *during learning* using the framework of rank-constrained classifiers [15, 10, 21]. We simultaneously learn a separable and steerable basis, together with the steering coefficients, using structural SVMs [20].

Our steerable representations can reduce the number of



(a) Changes in part viewpoint

(b) Vocabulary of parts

(c) Steerable basis

Figure 1: One needs large part vocabularies to capture variations in appearance due to viewpoint (**a**), among other factors. We approximate a large vocabulary of thousands of part templates (**b**) as linear combinations of a small set of basis parts (**c**). We show how can learn a steerable, separable basis together with the steering coefficients using rank-constrained structural SVMs. This reduces the number of model parameters by orders of magnitude, simplifying learning and increasing run-time speed.

model parameters by orders of magnitude, greatly simplifying both learning and run-time computation. Limiting the degrees of freedom reduces overfitting, which is useful for learning large part vocabularies from limited training data. Our steerable basis also represents a novel form of multi-task learning, in that the models learned for a set of object categories share information between them.

Since the current bottleneck of most part models is the computation of local filter scores, our steerable models improve run-time performance, often by an order of magnitude. This approaches recent speed ups obtained by cascaded [4] or coarse-to-fine [13] matching algorithms, but our performance gains come from an underlying change in the representation during learning, rather than an approximation that is made after a model is learned. For example, our efficiency gains still hold for high-recall detection, while other matching algorithms may slow down because one has to enumerate all candidate windows in an image. Because the efficiency gains are orthogonal, our steerable models can be combined with such matching algorithms to yield extremely lightweight and efficient detectors.

We learn steerable part models for three state-of-the-art systems for object detection [6] , articulated pose estimation [22], and facial landmark estimation [24]. In all three cases, we are able to achieve a near-10X reduction in parameters with little or no loss in performance. All of our systems are based on mixtures of parts, and so can be tuned for large or small vocabularies. When we compare these tuned systems with a steerable model with an equivalent number of parameters, our steerable models always significantly dominate in performance.

2. Related work

Our representation can be seen an approach to sharing parts. Parts are typically shared in a discrete fashion; for example, a single template for a wheel part may be shared across multiple view-based mixture models [19, 12] or within a compositional grammar of vehicles [23, 7]. In particular, [12] learns coefficients which calibrate parts that are shared across sub-category mixtures. However, because parts may look different under different viewpoints and compositions, we share a *linear subspace* rather than a fixed template, letting a small number of basis filters generate a large, near-continuous range of part appearances.

Our approach is inspired by classic work on low-level vision, including steerable filters [8], separable filters [1], and filter banks [1]. Our work is most similar to the framework of [11], who use an SVD to efficiently represent a large set of oriented and scaled filters. We follow a similar approach, but embed the rank restriction within the learning of the classifier, allowing us to discriminatively learn both a basis and reconstruction coefficients. Following the terminology of [11], we refer to these as a steerable basis and steering coefficients.

Our approach for rank-constrained learning is based on a recent collection of papers [15, 10, 21]. We follow the formalism of [15], who formulate a rank-constrained linear model as a bilinear model. In our case, given a fixed basis, our model is linear in the steering coefficients, and vice versa. [15] point out that convex learning algorithms for linear classifiers (such as SVMs) can be generalized to bi-convex learning algorithms that can be optimized with coordinate descent. Each step of the coordinate descent reduces to a standard SVM problem. In this work, we demonstrate such tools can be extended to simultaneously learn steerable and separable part filters using structural SVMs.

Any recognition algorithm that relies on a large vocabulary of linearly-scored templates would benefit from our approach. Some popular examples include poselets [2], visual phrases [17], and latent part models [6] . We focus on three illustrative examples. We use part-based model of [6] to illustrate part steering across object categories and subcategories. We use the recent articulated model of [22] to illustrate steering across in-plane orientations, useful for modeling articulation. We use the multiview model of [24] to illustrate steering across out-of-plane orientations, useful for modeling viewpoint variation in faces. Based on standard benchmark evaluation, all methods represent the state-of-the-art methods in human pose estimation, face pose/landmark estimation, and object detection. Our work provides a 10-100X reduction in model size, considerably simplifying learning and increasing run-time efficiency.

3. Steerable basis model

We begin by describing a general form of linearly-parameterized part models with mixtures, amenable to our steerable representation. The part models described in [6, 22, 24] can all be seen as instances of this general form.

Let $l_i = (x_i, y_i)$ be the pixel location of part i, and let t_i be the mixture component of part i. These mixture components may span different in-plane orientations (as in [22], out-of-plane orientations (as in [24]), or semantic subcategories of an object class (as in [6]). Given an image I, we score a collection of part locations $l = \{l_i\}$ and mixtures $t = \{t_i\}$ as:

$$\text{score}(I, l, t) = \Big[\sum_{i}^{m} w_i^{t_i} \cdot \phi_a(I, l_i) \Big] + w_s \cdot \phi_s(l, t) \quad (1)$$

where $\phi_a(I, l_i)$ is an appearance feature vector (e.g., HOG descriptor [3]) extracted from pixel location l_i in image I. The first term in (1) is an appearance model that computes the local score of placing filter w_i, tuned for mixture t_i, at location l_i.

Linear shape: The second term is a shape "prior" that favors particular configurations of part locations and mix-

tures. For our purposes, we are agnostic as to form of this function so long as it is linearly parameterized and there exist tractable algorithms for computing the best scoring configuration $\text{argmax}_{l,t}\,\text{score}(I,l,t)$. For example, [22] encodes springs between pairs of parts and co-occurrence priors between particular mixtures, where spring rest position, spring rigidity, and co-occurrence biases are all encoded linearly in w_s. When pairwise relations are restricted to a tree, the best-matching configuration can be computed with dynamic programming. Usually, there are very few parameters in the prior term compared to the appearance terms. We will describe linear subspace methods for approximating the high-dimensional appearance terms.

Bilinear appearance: Assume that all part filters are identical in size, each requiring n_d parameters. If they are not, one can zero-pad templates to achieve this. We would like to write each filter as a linear combination of n_s basis filters:

$$w_i^{t_i} = \sum_{j=1}^{n_s} b_j s_{ij}^{t_i} \tag{2}$$

where b_j is a basis filter of size n_d and $s_{ij}^{t_i}$ is the scalar steering coefficient specific to each part mixture $w_i^{t_i}$. Plugging (2) into (1), we can see that our scoring function is now *bilinear*; if we fix the basis b, it is linear in the coefficients s and vice versa. We now make this connection explicit by introducing matrix notation for (1) and defining $z = (l,t)$:

$$\text{score}(I,z) = \text{Tr}(W_a^T \Phi_a(I,z)) + w_s^T \phi_s(z) \tag{3}$$

where $W_a \in \mathbb{R}^{n_d \times n_p}$ is the filter bank (in column-wise order). Each column of W_a represents a single part filter with n_d parameters. We write n_p for the total size of the part vocabulary, including all mixtures of all parts. $\Phi_a(I,z) \in \mathbb{R}^{n_d \times n_p}$ is a sparse matrix of local appearance features extracted from location l, with nonzero entries corresponding to the active mixtures t. Finally, Tr() is the standard trace operator, where we use the property that $\text{Tr}(A^T B) = A(:)^T B(:)$ following Matlab notation. We can then write W_a as a product of a basis and steering coefficient matrix:

$$W_a = BS^T \quad \text{where} \quad B \in \mathbb{R}^{n_d \times n_s}, S \in \mathbb{R}^{n_p \times n_s} \tag{4}$$

where each column of B is a basis filter and each row of S specifies the steering coefficients for a particular part mixture. We see that our steerable model can be interpreted as a rank n_s constraint on the filter bank W_a. Plugging in (4) into (3), we get:

$$\text{score}(I,z) = \text{Tr}(SB^T \Phi_a(I,z)) + w_s^T \phi_s(z) \tag{5}$$

The new score is bilinear in its parameters (B,S,w_s); if we fix the steerable basis B, the score is linear in the coefficients S and shape parameters w_s. If we fix S, the score is linear in B and w_s. This suggests that we can generalize convex algorithms for learning parameters (such as SVMs) to biconvex algorithms that iterate learning one set of parameters holding the others fixed.

4. Learning

We now describe a method for learning steerable models given supervised data, where part locations and mixture types, are known for all positive training examples. When these variables are not known, [6] describes a coordinate descent algorithm that iterates between (1) learning model parameters assuming such latent variables are known and (2) updating latent variables given the learned model parameters. For weakly supervised datasets, our learning algorithm can be applied *within* step (1). If all labels are given in a supervised framework (as is the case for [22] and [24]), then one can directly apply the approach here.

Given labeled positive examples $\{I_n, z_n\}$ and negative examples $\{I_n\}$ we can write our max-margin learning formulation as:

$$\begin{aligned} L(w) = &\frac{1}{2} w^T w + C \sum_n \max_{z \in Z_n} [0, 1 - y_n w^T \phi(I_n, z)] \\ &Z_n = \{z_n\} \quad \forall n \quad \text{s.t.} \quad y_n = 1 \\ &Z_n = \{\text{unrestricted}\} \quad \forall n \quad \text{s.t.} \quad y_n = -1 \end{aligned} \tag{6}$$

where parameters and features in (1) are vectorized and concatenated to form vectors w and $\phi(I_n,z)$. The first term is a standard regularizer. The second term is a cumulative loss over training examples. For each positive training example I_n, if the score of the given labels z_n is greater than 1, the loss evaluates to 0. Else the loss is set to the difference (the slack). For each negative training example, one searches all possible locations and mixture types z for the highest scoring configuration. If its score is less than -1, the loss evaluates to 0; else the loss is set to the difference.

The above formulation is equivalent to the convex inner loop of a latent SVM [6]. It can also be written as a structural SVM, for which many excellent large-scale solvers exist [6, 20]. To optimize the above, we rewrite it as a quadratic objective with linear constraints, minimizing it with the dual-coordinate descent quadratic program (QP) solver of [22].

We can rewrite (6) in matrix form as:

$$\begin{aligned} L(W_a, w_s) = &\frac{1}{2} \text{Tr}(W_a^T W_a) + \frac{1}{2} w_s^T w_s + \\ &C \sum_n \max_{z \in Z_n} [0, 1 - y_n \Big(\text{Tr}(W_a^T \Phi_a(I,z)) + w_s^T \phi_s(z) \Big)] \end{aligned} \tag{7}$$

This means that any equation that can be written as (7) can be solved by an off-the-shelf structured SVM solver. By substituting a steerable basis B and coefficient matrix S for

the filter bank W_a, we can write the objective function as $L(B, S, w_s)$. With the following key substitutions,

$$\mathrm{Tr}(W_a^T W_a) = \mathrm{Tr}(SB^T BS^T) \quad (8)$$

$$\mathrm{Tr}(W_a^T \Phi_a) = \mathrm{Tr}(SB^T \Phi_a) = \mathrm{Tr}(B^T \Phi_a S) \quad (9)$$

we can rewrite (7) as:

$$L(B, S, w_s) = \frac{1}{2}\mathrm{Tr}(SB^T BS^T) + \frac{1}{2} w_s^T w_s + \quad (10)$$
$$C \sum_n \max_{z \in Z_n} [0, 1 - y_n \Big(\mathrm{Tr}(B^T \Phi_a(I, z) S) + w_s^T \phi_s(z) \Big)]$$

The above function is no longer convex in its arguments. However, by freezing the steering coefficients S, the above function can be written as a convex function:

$$L(\tilde{B}, w_s) = \frac{1}{2}\mathrm{Tr}(\tilde{B}^T \tilde{B}) + \frac{1}{2} w_s^T w_s + \quad (11)$$
$$C \sum_n \max_{z \in Z_n} [0, 1 - y_n \Big(\mathrm{Tr}(\tilde{B}^T \tilde{\Phi}_a(I_n, z_n)) + w_s^T \phi_s(z) \Big)]$$
$$\text{where} \quad \tilde{B} = BA^{\frac{1}{2}}, \quad \tilde{\Phi}_a = \Phi_a S A^{-\frac{1}{2}}, \quad A = S^T S$$

(11) is equivalent in structure to (7); hence it is convex and can be optimized with an off-the-shelf structured SVM solver. Given a solution, we can recover the final steerable basis $B = \tilde{B} A^{-\frac{1}{2}}$. Note that $A = S^T S$ is $n_s \times n_s$ matrix that will in general be invertible given for $n_s \ll n_p$ (e.g., a small number of basis filters compared to a large part vocabulary). One can easily show a similar convex formulation for optimizing $L(S, w_s)$ given a fixed steerable basis B. This makes the overall formulation from (10) biconvex in its arguments, amenable to coordinate descent algorithms for minimization [15]. Specifically, given some initial steerable basis B^*, iterate the following steps using a structured SVM solver:

1. $(S^*, w_s^*) = \mathrm{argmin}_{S, w_s} L(B^*, S, w_s)$
2. $(B^*, w_s^*) = \mathrm{argmin}_{B, w_s} L(B, S^*, w_s)$

Initialization: In practice, to initialize B^*, we first independently learn a filter for each part with a standard linear SVM. This is typically inexpensive and parallelizable. We then apply a rank-n_s SVD to this set to estimate an initial B^*.

Latent alignment: A traditional difficulty with subspace methods is that of alignment; if patches are not aligned well, then low-rank approximations will tend to be very blurred. By iterating both over our steerable parameters (S, B, w_s) and latent configuration variables z, our learning algorithm can re-align parts to better match our steerable basis. Hence, even for fully-supervised datasets where part locations z are known, we allow for small latent translations that re-align parts as we learn a steerable basis.

5. Steerability and separability

Thus far we have made no assumption on the form of each basis filter, beyond the fact that it contains n_d parameters. We now augment our model to enforce the fact that each basis filter is separable. One can model each n_d-length basis filter as a $n_y \times n_x \times n_f$ tensor, encoding a spatial neighborhood of $n_y \times n_x$ cells, with n_f orientation features extracted from each cell. A fully-separable filter can be written as a rank-1 tensor, or a product of three one-dimensional vectors. For simplicity, we focus on separability in one dimension. To do so, let us reshape each basis filter b_j from (2) into a $n_{xy} \times n_f$ matrix B_j that is restricted to be low rank:

$$B_j = \sum_{k=1}^{n_k} c_{jk} f_{jk}^T \quad \text{where} \quad c_{jk} \in \mathbb{R}^{n_{xy} \times 1}, f_{jk} \in \mathbb{R}^{n_f \times 1}$$

where $n_k = 1$ corresponds to the fully separable case. We refer to c_{jk} as the spatial basis and f_{jk} as the feature basis. Combining this with (2), we can write each part filter as:

$$W_i^{t_i} = \sum_{j=1}^{n_s} \sum_{k=1}^{n_k} c_{jk} f_{jk}^T s_{ij}^{t_i} \quad \text{where} \quad W_i^{t_i} \in \mathbb{R}^{n_{xy} \times n_f}$$

When plugging this expression back into (3), we see that the overall score function is now *multilinear* in its parameters. By fixing two sets of its parameters (say the feature basis and steering coefficients), it is simultaneously linear in the third (the spatial basis) and the spatial parameters w_s. The resulting learning problem is *multiconvex*, amenable to coordinate descent where each step corresponds to solving a problem of the form from (11), derived by holding two parameters fixed and solving for the third. Again, this convex program can be solved with an off-the-shelf structural SVM solver. We omit the straightforward but cluttered equations for lack of space.

One can combine the two approaches by learning a "shared basis" of separability. For example, one could force all basis filters B_j to share the same feature basis:

$$f_{jk} = f_k$$

One can then interpret f_k as vectors that span a generic feature basis used by all basis filters. We consider this form of separability in our experiments, as it considerably reduces the number of parameters even further.

6. Multi-category learning

Current category-level models are trained and detected independently for each object category [4]. This will clearly not scale to tens of thousands of categories. An open question is how to share structure across such models, both for purposes of increased regularization and computational savings. We show that our steerable framework provides one natural mechanism for sharing.

Assume we wish to train M category-specific models, but wish to learn a shared steerable/separable vocabulary across all categories. Given positive and negative training examples for each category, we can write all M learning problems as one optimization:

$$L_{\text{multi}}(W_a^1 \dots W_a^M, w_s^1 \dots w_a^M) = \sum_{m=1}^{M} L(W_a^m, w_s^m)$$

where $L(W_a^m, w_s^m)$ is defined as (7). As written, the above problem can be optimized over each category-specific model independently. Instead, we share structure across categories by restricting all models to share the same basis filters:

$$W_a^m = BS_m^T$$

The previously described coordinate descent algorithm applies to this multi-category formulation as well. Given a fixed basis B, it is straightforward to optimize (S_m, w_s^m) with M independent structural SVM problems. Given the learned steering coefficients S_m for each category, a single steerable basis is learned by solving a single structural SVM problem using all the training data. We omit the straightforward but cluttered equations for lack of space.

7. Computational savings

Given a learned model, at run-time, we want to maximize score(I, z) in (1) over all part positions p and mixture types t. As stated previously, we assume that the shape prior $w_s \cdot \phi_s(p, t)$ factors into a tree model, amenable to dynamic programming. Many practitioners have observed that the bottleneck of the dynamic programming is the computation of the local part scores, for all parts in the vocabulary at all image locations [4, 13]. We analyze the computational savings afforded by steerable and separable representations.

Inference: Following our existing notation, assume we have a vocabulary of n_p parts of dimension $n_{xy} \times n_f$. Given an image with N cell locations, naive computation of the local score will take $O(Nn_{xy}n_f n_p)$. One can show that given a shared feature basis of n_k dimensions and n_s basis filters, the local score computation is dominated by the convolution of basis filters, requiring $O(Nn_{xy}n_k n_s)$. For typical parameter values, this results in a 10X to 100X reduction in the number of operations.

Learning: The above savings for inference also speed up learning, since one must run the inference algorithm to collect support vectors. Our steerable model also decreases storage costs by a similar amount. Support vectors will be 10X to 100X smaller since we need only optimize/store a small fraction of the appearance features during each step of coordinate descent. This is practically significant since many models require a large number of support vectors, approaching memory limits of typical hardware. Hence our factored representation allows us to learn significantly larger models.

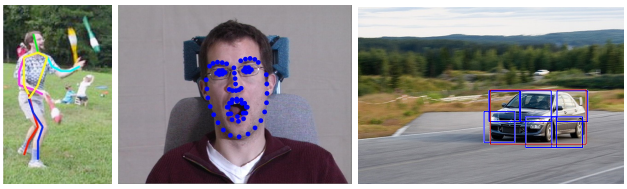


Figure 2: We show results of applying our steerable/separable representation to three diverse but well-benchmarked applications of articulated pose estimation (on PARSE), facial landmark estimation (on MultiPIE), and object detection (on PASCAL VOC 2007).

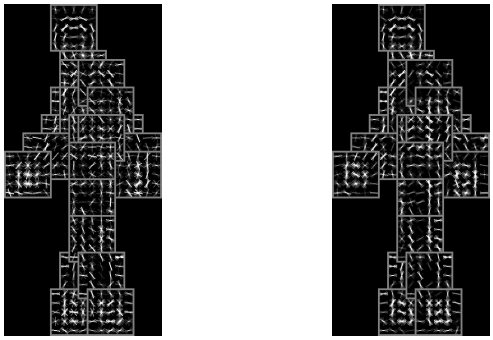
8. Experiments

We apply our method on three recognition tasks; articulated pose estimation, facial pose estimation/landmark estimation, and object detection (Fig. 2). We use standard benchmark datasets, evaluation protocols, and compare results with state-of-the-art methods (as reported on these datasets).

8.1. Articulated human pose estimation

The articulated part model of [22] is a pictorial structure defined on a part vocabulary of size $n_p = 138, n_{xy} = 25, n_f = 32$. These parts represent different in-plane orientations of articulated body parts, and so are natural candidates for a steerable representation. We train and test the model on the Image Parse dataset [16], following the standard evaluation protocol and criteria (PCP score). We visualize the original model from [22], and its reconstruction under our steerable representation in Fig. 3. We show a subset of the 138-part vocabulary and the learned steerable filters in Fig. 1.

Fig. 4 compares our results with [22] for various parameter settings. Notably, the baseline model can be tuned for different number of parameters by varying the number of discrete orientations modeled at each part, from 1 to 6. We show that, given a fixed number of parameters, we always obtain a better model using steerable parts. We obtain similar (and even slightly better) PCP performance with a 15X reduction in the total number of parameters, and even obtain reasonable performance for a 100X reduction in model size. These translate to roughly similar improvements in run-time speed, given the single-threaded public matlab/mex implementation of [22]. For example, per-image running time decreases from 6 seconds to almost 0.5 seconds for our $n_s = 20, n_k = 8$ model. We show part-specific localization accuracy in Table 1. Finally, we show performance as a function of the number of iterations of coordinate descent in Fig. 5. We see that the initial steerable basis, though



(a) Baseline ($n_p = 138, n_f = 32$) (b) Our model ($n_s = 20, n_k = 8$)

Figure 3: We show the original articulated human model from [22] in (**a**) and our steerable/separable reconstruction in (**b**). Our model looks and performs similar, but is roughly 15X smaller and faster at inference time.

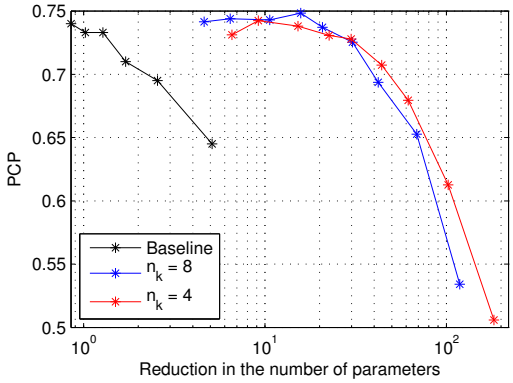


Figure 4: We compare the performance of the articulated pose model of [22] with our steerable/separable variant for equivalent number of parameters (on the PARSE benchmark). One can reduce the number of parameters in [22] by limiting the number of oriented mixtures of each part (the black baseline), or by exploiting our steerable representation. We plot two curves obtained by varying the number of basis filters n_s for different feature dimensions n_k. Our model always dominates for an equivalent number of parameters, and even achieves state-of-the-art performance with a 15X reduction in parameters. We also achieve reasonable performance with 100X reduction in parameters.

initialized with care (as described in Sec. 4) considerably improves after a full iteration of coordinate descent.

8.2. Facial pose estimation / landmark localization

The view-based part model of [24] appears to be current state-of-the-art system for pose estimation and landmark localization, and given by evaluation on the MultiPIE benchmark [9]. This model defines a mixtures-of-trees part facial landmark model, where each mixture encodes an out-of-plane change in orientation. The best performance reported in [24] is given by a model which defines a separate part

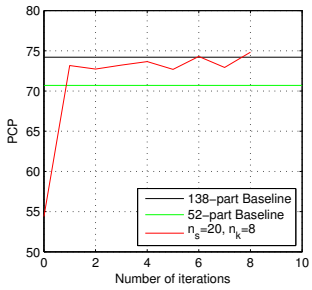
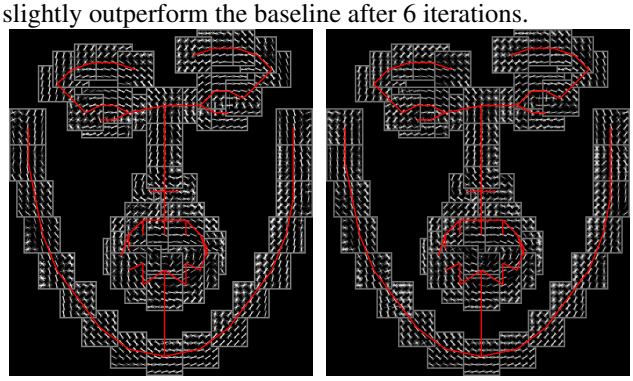


Figure 5: We plot PCP performance over iterations of coordinate descent. Iteration 0 corresponds to initialing both a steerable basis and feature subspace with an SVD of part filters trained independently with SVMs. We see a significant improvement in results in a single iteration, and even slightly outperform the baseline after 6 iterations.



(a) Baseline (n_p=1050, n_f=32) (b) Our model (n_s=93, n_k=8)

Figure 6: We show one of views from the multi-view facial model from [24] in (**a**) and our steerable/separable reconstruction in (**b**). Tree-structured spatial constraints between parts are drawn as red lines. Our model looks and performs similar, but is roughly 7X smaller and faster at inference time.

filter for each out-of-plane orientation, resulting in a large vocabulary of size $n_p = 1050, n_{xy} = 25, n_f = 32$.

Table 2 and Fig. 7 compare our results with [24] for various parameter settings. Notably, the baseline model can be tuned for different number of parameters by varying the number of out-of-plane orientations modeled at each part. Tuning both our model and the baseline for a 7X reduction in parameters, our model dominates in both pose estimation and landmark localization accuracy. Our model even slightly improves upon the state-of-the-art pose estimation reported in [24], and still produces reasonable performance for a 22X reduction in model size.

8.3. Sharing across object categories

We now present results for steerable variants of the popular deformable part model of [6]. We show results on the PASCAL VOC 2007 testset using standard evaluation criteria. Notably, we apply the multi-category steerable representation presented in Sec.6. For simplicity, we only ap-

Method	Reduction in # params	# basis n_s	Feature dimension n_k	Torso	Head	Upper legs	Lower legs	Upper arms	Lower arms	Total
138-part baseline[22]	1	-	-	96.6	93.2	82.0	74.1	72.9	47.3	74.2
52-part baseline[22]	2.7	-	-	95.6	92.2	82.2	72.4	68.5	36.3	70.7
Our Model	15.7	20	8	97.6	92.2	81.2	74.4	74.9	48.8	74.8
Our Model	22.6	20	4	97.1	90.2	82.0	75.4	72.2	42.2	73.1

Table 1: We compare articulated pose estimation results with [22], which reports state-of-the-art results on the PARSE benchmark [16]. We use the standard evaluation criteria of PCP. We compare results to the baseline model [22] tuned for different numbers of orientation mixtures. We report the results for different numbers of basis filters n_s and different feature dimensions n_k. We achieve almost the same total and per part results with over 10 times reduction in the number of parameters and running time. Our model even slightly outperforms the baseline for some parts.

Method	Reduction in # params	# basis n_s	Subspace dimension n_k	Accuracy of exact pose estimation	Localization error (mse)
1050-part baseline[24]	1	-	-	91.4	0.0234
146-part baseline[24]	7.2	-	-	82.0	0.0256
99-shared baseline[24]	10.6	-	-	81.5	0.0281
Our Model	7.2	93	8	91.6	0.0236
Our Model	22.2	30	8	89.3	0.0247
Our Model	24.3	30	4	89.9	0.0256

Table 2: We compare our results in facial detection, pose estimation, and landmark localization with the baseline in [24]. We compare against baseline models with different amounts of sharing. We show the results for different number of basis filters n_s and different subspace dimensions n_k. We achieve almost the same performance, in both pose estimation and landmark localization, with a 7-20X reduction in model size.

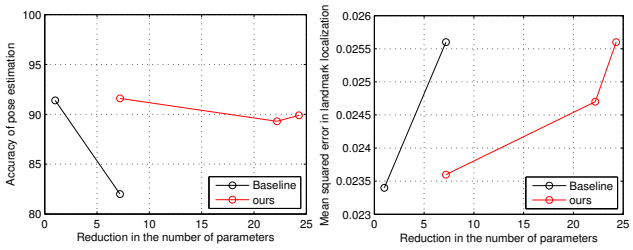


Figure 7: Facial pose estimation and landmark localization on MultiPIE. We compare to two versions of the baseline model from [24], tuned for 1 mixture per part or 7 view-based mixtures per part. On the left, we plot pose estimation accuracy (higher is better). On the right, we plot landmark localization error (lower is better). For an equivalent number of parameters, our model greatly outperforms the baseline. We also closely match baseline performance with a 24X reduction in model size.

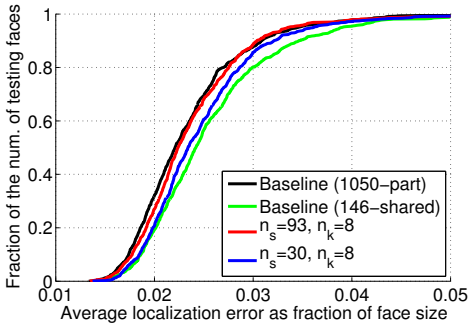


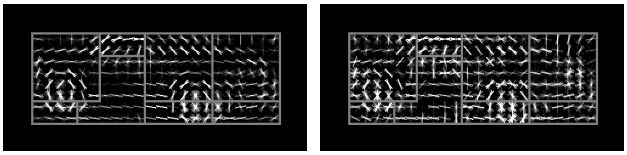
Figure 8: We evaluate landmark localization accuracy with cumulative error plots. We plot the fraction of test images for which landmark localization error is below a given value. We compare against baseline models with vocabularies of 1050 and 146 parts. Our model gives roughly equivalent performance with a 7-20X reduction in model size.

ply our steerable representation on the part filters and not the root filters, since the former are all equivalent in size. Across all categories, the size of the part vocabulary can be written as $n_p = 480, n_{xy} = 36, n_f = 32$. We explore a steerable model with $n_s = 60, n_f = 32$, since we found that a shared feature basis hurts in the multi-category scenario. Our models are 3X smaller and faster with a near equivalent performance (Table 3 and Fig. 9). Our parameter reduction would be greater if we also implemented steerable root filters; we are currently pursuing extensions for such variable-size filter banks.

Conclusion: We describe a method for learning steerable deformable part models, based on the observation that part templates can be written as linear filter banks. We show how can leverage existing SVM-solvers to learn steerable representations using rank-constraints. We demonstrate impressive results on three diverse problems in recognition, showing improvements up to 10X-100X in size and speed. When we compare our steerable models with baselines tuned for equivalent sizes, our models always dom-

Category	plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	
voc-rel4[5]	29.6	57.3	10.1	17.1	25.2	47.8	55.0	18.4	21.6	24.7	
Our Baseline	27.1	57.1	10.2	13.9	22.5	47.3	52.2	17.4	17.8	23.8	
Our Model	29.7	56.6	10.2	15.3	23.1	48.7	53.8	15.7	19.9	22.2	
Category	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	total
voc-rel4[5]	23.3	11.2	57.6	46.5	42.1	12.2	18.6	31.9	44.5	40.9	31.8
Our Baseline	20.4	6.8	56.1	43.5	42.3	12.0	18.5	32.5	39.0	39.7	30.0
Our Model	20.5	4.3	56.0	46.0	40.4	12.3	18.6	30.1	40.4	41.4	30.3

Table 3: Average precision for different object categories in PASCAL 2007 dataset. The first row contains the results reported in the released code of [5] without any post-processing. We reimplemented the code to allow for easier modification. Our reimplementation is shown in the second row. The third row is the steerable variant of our reimplementation, tuned for $n_s = 60$ and 3X reduction in the number of parameters. Our performance slightly increases while yielding a smaller and faster model.



(a) Baseline (n_p=480) (b) Our model (n_s=60)

Figure 9: On the left, we show the result of our implementation of the `car` model from [6]. On the right, we show our learned model reconstructed from 60 steerable basis filters, shared across all 20 object categories. Our model looks and performs similar, but is 3X smaller and faster at run-time.

inate in performance, suggesting that they are more natural representations. Finally, we demonstrate that steerable structure can be shared across different object categories.

Acknowledgements: Funding for this research was provided by NSF Grant 0954083 and ONR-MURI Grant N00014-10-1-0933.

References

[1] E. Adelson and J. Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2(2), 1985.

[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, pages 1365–1372. IEEE, 2009.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005.

[4] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248. IEEE, 2010.

[5] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://www.cs.brown.edu/ pff/latent-release4/.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE TPAMI*, 32(9), 2010.

[7] S. Fidler, M. Boben, and A. Leonardis. Learning hierarchical compositional representations of object structure. In *Object categorization: computer and human perspectives*. Cambridge, 2009.

[8] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE TPAMI*, 13(9):891–906, 1991.

[9] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 28(5), 2010.

[10] I. Kotsia and I. Patras. Support tucker machines. In *CVPR*, 2011.

[11] R. Manduchi, P. Perona, and D. Shy. Efficient deformable filter banks. *Signal Processing, IEEE Transactions on*, 46(4):1168–1173, 1998.

[12] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *CVPR*, pages 1513 –1520, june 2011.

[13] M. Pedersoli, A. Vedaldi, and J. Gonzàlez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011.

[14] P. Perona. Deformable kernels for early vision. *IEEE TPAMI*, 17(5):488–499, 1995.

[15] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Bilinear Classifiers for Visual Recognition. *NIPS*, 2009.

[16] D. Ramanan. Learning to parse images of articulated bodies. *NIPS*, 19:1129, 2007.

[17] M. Sadeghi and A. Farhadi. Recognition using visual phrases. In *CVPR*, 2011.

[18] J. Sivic and A. Zisserman. Video google: Efficient visual search of videos. *Toward Category-Level Object Recognition*, 2006.

[19] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE TPAMI*, 29(5):854–869, 2007.

[20] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2):1453, 2006.

[21] L. Wolf, H. Jhuang, and T. Hazan. Modeling appearances with low-rank svm. In *CVPR*, 2007.

[22] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *CVPR*, 2011.

[23] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille. Part and appearance sharing: Recursive compositional models for multi-view multi-object detection. *Pattern Recognition*, 2010.

[24] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark estimation in the wild. *CVPR*, 2012.