A Discriminatively Trained, Multiscale, Deformable Part Model

Pedro Felzenszwalb
University of Chicago
pff@cs.uchicago.edu

David McAllester
Toyota Technological Institute at Chicago
mcallester@tti-c.org

Deva Ramanan
TTI-C and UC Irvine
dramanan@ics.uci.edu

Abstract

This paper describes a discriminatively trained, multiscale, deformable part model for object detection. Our system achieves a two-fold improvement in average precision over the best performance in the 2006 PASCAL person detection challenge. It also outperforms the best results in the 2007 challenge in ten out of twenty categories. The system relies heavily on deformable parts. While deformable part models have become quite popular, their value had not been demonstrated on difficult benchmarks such as the PASCAL challenge. Our system also relies heavily on new methods for discriminative training. We combine a margin-sensitive approach for data mining hard negative examples with a formalism we call latent SVM. *A latent SVM, like a hidden CRF, leads to a non-convex training problem. However, a latent SVM is semi-convex and the training problem becomes convex once latent information is specified for the positive examples. We believe that our training methods will eventually make possible the effective use of more latent information such as hierarchical (grammar) models and models involving latent three dimensional pose.*

1. Introduction

We consider the problem of detecting and localizing objects of a generic category, such as people or cars, in static images. We have developed a new multiscale deformable part model for solving this problem. The models are trained using a discriminative procedure that only requires bounding box labels for the positive examples. Using these models we implemented a detection system that is both highly efficient and accurate, processing an image in about 2 seconds and achieving recognition rates that are significantly better than previous systems.

Our system achieves a two-fold improvement in average precision over the winning system [5] in the 2006 PASCAL person detection challenge. The system also outperforms the best results in the 2007 challenge in ten out of twenty object categories. Figure 1 shows an example detection obtained with our person model.

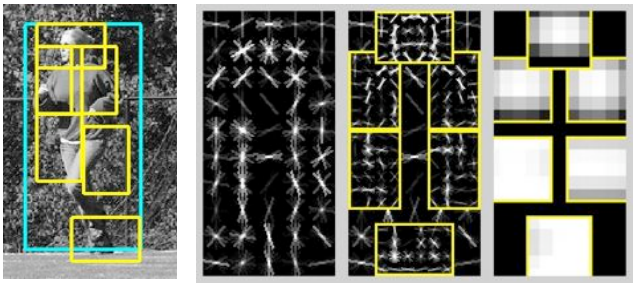

Figure 1. Example detection obtained with the person model. The model is defined by a coarse template, several higher resolution part templates and a spatial model for the location of each part.

The notion that objects can be modeled by parts in a deformable configuration provides an elegant framework for representing object categories [1–3, 6, 10, 12, 13, 15, 16, 22]. While these models are appealing from a conceptual point of view, it has been difficult to establish their value in practice. On difficult datasets, deformable models are often outperformed by "conceptually weaker" models such as rigid templates [5] or bag-of-features [23]. One of our main goals is to address this performance gap.

Our models include both a coarse global template covering an entire object and higher resolution part templates. The templates represent histogram of gradient features [5]. As in [14, 19, 21], we train models discriminatively. However, our system is semi-supervised, trained with a max-margin framework, and does not rely on feature detection. We also describe a simple and effective strategy for learning parts from weakly-labeled data. In contrast to computationally demanding approaches such as [4], we can learn a model in 3 hours on a single CPU.

Another contribution of our work is a new methodology for discriminative training. We generalize SVMs for handling latent variables such as part positions, and introduce a new method for data mining "hard negative" examples during training. We believe that handling partially labeled data is a significant issue in machine learning for computer vision. For example, the PASCAL dataset only specifies a bounding box for each positive example of an object. We treat the position of each object part as a latent variable. We

also treat the exact location of the object as a latent variable, requiring only that our classifier select a window that has large overlap with the labeled bounding box.

A latent SVM, like a hidden CRF [19], leads to a non-convex training problem. However, unlike a hidden CRF, a latent SVM is semi-convex and the training problem becomes convex once latent information is specified for the positive training examples. This leads to a general coordinate descent algorithm for latent SVMs.

System Overview Our system uses a scanning window approach. A model for an object consists of a global "root" filter and several part models. Each part model specifies a spatial model and a part filter. The spatial model defines a set of allowed placements for a part relative to a detection window, and a deformation cost for each placement.

The score of a detection window is the score of the root filter on the window plus the sum over parts, of the maximum over placements of that part, of the part filter score on the resulting subwindow minus the deformation cost. This is similar to classical part-based models [10, 13]. Both root and part filters are scored by computing the dot product between a set of weights and histogram of gradient (HOG) features within a window. The root filter is equivalent to a Dalal-Triggs model [5]. The features for the part filters are computed at twice the spatial resolution of the root filter. Our model is defined at a fixed scale, and we detect objects by searching over an image pyramid.

In training we are given a set of images annotated with bounding boxes around each instance of an object. We reduce the detection problem to a binary classification problem. Each example x is scored by a function of the form, $f_\beta(x) = \max_z \beta \cdot \Phi(x, z)$. Here β is a vector of model parameters and z are latent values (e.g. the part placements). To learn a model we define a generalization of SVMs that we call *latent variable SVM* (LSVM). An important property of LSVMs is that the training problem becomes convex if we fix the latent values for positive examples. This can be used in a coordinate descent algorithm.

In practice we iteratively apply classical SVM training to triples $(\langle x_1, z_1, y_1\rangle, \ldots, \langle x_n, z_n, y_n\rangle)$ where z_i is selected to be the best scoring latent label for x_i under the model learned in the previous iteration. An initial root filter is generated from the bounding boxes in the PASCAL dataset. The parts are initialized from this root filter.

2. Model

The underlying building blocks for our models are the Histogram of Oriented Gradient (HOG) features from [5]. We represent HOG features at two different scales. Coarse features are captured by a rigid template covering an entire detection window. Finer scale features are captured by part templates that can be moved with respect to the detection window. The spatial model for the part locations is equivalent to a star graph or 1-fan [3] where the coarse template serves as a reference position.

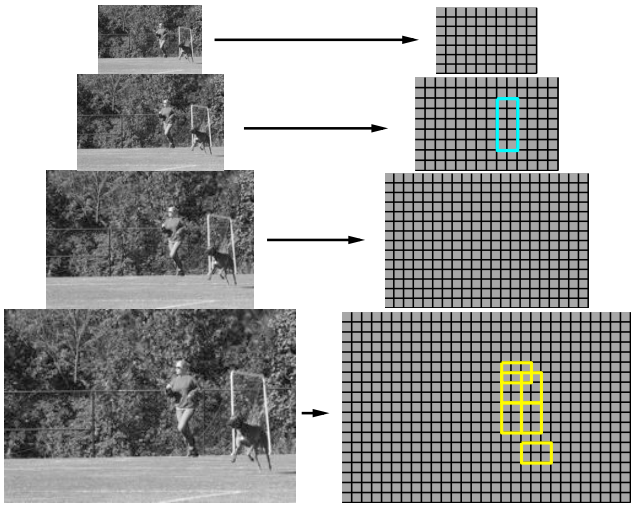

Figure 2. The HOG feature pyramid and an object hypothesis defined in terms of a placement of the root filter (near the top of the pyramid) and the part filters (near the bottom of the pyramid).

2.1. HOG Representation

We follow the construction in [5] to define a dense representation of an image at a particular resolution. The image is first divided into 8x8 non-overlapping pixel regions, or *cells*. For each cell we accumulate a 1D histogram of gradient orientations over pixels in that cell. These histograms capture local shape properties but are also somewhat invariant to small deformations.

The gradient at each pixel is discretized into one of nine orientation bins, and each pixel "votes" for the orientation of its gradient, with a strength that depends on the gradient magnitude at that pixel. For color images, we compute the gradient of each color channel and pick the channel with highest gradient magnitude at each pixel. Finally, the histogram of each cell is normalized with respect to the gradient energy in a neighborhood around it. We look at the four 2×2 blocks of cells that contain a particular cell and normalize the histogram of the given cell with respect to the total energy in each of these blocks. This leads to a 9×4 dimensional vector representing the local gradient information inside a cell.

We define a HOG feature pyramid by computing HOG features of each level of a standard image pyramid (see Figure 2). Features at the top of this pyramid capture coarse gradients histogrammed over fairly large areas of the input image while features at the bottom of the pyramid capture finer gradients histogrammed over small areas.

2.2. Filters

Filters are rectangular templates specifying weights for subwindows of a HOG pyramid. A w by h filter F is a vector with $w \times h \times 9 \times 4$ weights. The score of a filter is defined by taking the dot product of the weight vector and the features in a $w \times h$ subwindow of a HOG pyramid.

The system in [5] uses a single filter to define an object model. That system detects objects from a particular class by scoring every $w \times h$ subwindow of a HOG pyramid and thresholding the scores.

Let H be a HOG pyramid and $p = (x, y, l)$ be a cell in the l-th level of the pyramid. Let $\phi(H, p, w, h)$ denote the vector obtained by concatenating the HOG features in the $w \times h$ subwindow of H with top-left corner at p. The score of F on this detection window is $F \cdot \phi(H, p, w, h)$.

Below we use $\phi(H, p)$ to denote $\phi(H, p, w, h)$ when the dimensions are clear from context.

2.3. Deformable Parts

Here we consider models defined by a coarse root filter that covers the entire object and higher resolution part filters covering smaller parts of the object. Figure 2 illustrates a placement of such a model in a HOG pyramid. The root filter location defines the detection window (the pixels inside the cells covered by the filter). The part filters are placed several levels down in the pyramid, so the HOG cells at that level have half the size of cells in the root filter level.

We have found that using higher resolution features for defining part filters is essential for obtaining high recognition performance. With this approach the part filters represent finer resolution edges that are localized to greater accuracy when compared to the edges represented in the root filter. For example, consider building a model for a face. The root filter could capture coarse resolution edges such as the face boundary while the part filters could capture details such as eyes, nose and mouth.

The model for an object with n parts is formally defined by a root filter F_0 and a set of part models $(P_1, \ldots, P_n)$ where $P_i = (F_i, v_i, s_i, a_i, b_i)$. Here F_i is a filter for the i-th part, v_i is a two-dimensional vector specifying the center for a box of possible positions for part i relative to the root position, s_i gives the size of this box, while a_i and b_i are two-dimensional vectors specifying coefficients of a quadratic function measuring a score for each possible placement of the i-th part. Figure 1 illustrates a person model.

A placement of a model in a HOG pyramid is given by $z = (p_0, \ldots, p_n)$, where $p_i = (x_i, y_i, l_i)$ is the location of the root filter when $i = 0$ and the location of the i-th part when $i > 0$. We assume the level of each part is such that a HOG cell at that level has half the size of a HOG cell at the root level. The score of a placement is given by the scores of each filter (the data term) plus a score of the placement of each part relative to the root (the spatial term),

$$\sum_{i=0}^{n} F_i \cdot \phi(H, p_i) + \sum_{i=1}^{n} a_i \cdot (\tilde{x}_i, \tilde{y}_i) + b_i \cdot (\tilde{x}_i^2, \tilde{y}_i^2), \quad (1)$$

where $(\tilde{x}_i, \tilde{y}_i) = ((x_i, y_i) - 2(x, y) + v_i)/s_i$ gives the location of the i-th part relative to the root location. Both $\tilde{x}_i$ and $\tilde{y}_i$ should be between -1 and 1.

There is a large (exponential) number of placements for a model in a HOG pyramid. We use dynamic programming and distance transforms techniques [9, 10] to compute the best location for the parts of a model as a function of the root location. This takes $O(nk)$ time, where n is the number of parts in the model and k is the number of cells in the HOG pyramid. To detect objects in an image we score root locations according to the best possible placement of the parts and threshold this score.

The score of a placement z can be expressed in terms of the dot product, $\beta \cdot \psi(H, z)$, between a vector of model parameters β and a vector $\psi(H, z)$,

$$\beta = (F_0, \ldots, F_n, a_1, b_1 \ldots, a_n, b_n).$$
$$\psi(H, z) = (\phi(H, p_0), \phi(H, p_1), \ldots \phi(H, p_n),$$
$$\tilde{x}_1, \tilde{y}_1, \tilde{x}_1^2, \tilde{y}_1^2, \ldots, \tilde{x}_n, \tilde{y}_n, \tilde{x}_n^2, \tilde{y}_n^2,).$$

We use this representation for learning the model parameters as it makes a connection between our deformable models and linear classifiers.

On interesting aspect of the spatial models defined here is that we allow for the coefficients (a_i, b_i) to be negative. This is more general than the quadratic "spring" cost that has been used in previous work.

3. Learning

The PASCAL training data consists of a large set of images with bounding boxes around each instance of an object. We reduce the problem of learning a deformable part model with this data to a binary classification problem. Let $D = (\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)$ be a set of labeled examples where $y_i \in \{-1, 1\}$ and x_i specifies a HOG pyramid, $H(x_i)$, together with a range, $Z(x_i)$, of valid placements for the root and part filters. We construct a positive example from each bounding box in the training set. For these examples we define $Z(x_i)$ so the root filter must be placed to overlap the bounding box by at least 50%. Negative examples come from images that do not contain the target object. Each placement of the root filter in such an image yields a negative training example.

Note that for the positive examples we treat both the part locations and the exact location of the root filter as latent variables. We have found that allowing uncertainty in the root location during training significantly improves the performance of the system (see Section 4).

3.1. Latent SVMs

A latent SVM is defined as follows. We assume that each example x is scored by a function of the form,

$$f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z), \quad (2)$$

where β is a vector of model parameters and z is a set of latent values. For our deformable models we define $\Phi(x, z) = \psi(H(x), z)$ so that $\beta \cdot \Phi(x, z)$ is the score of placing the model according to z.

In analogy to classical SVMs we would like to train β from labeled examples $D = (\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)$ by optimizing the following objective function,

$$\beta^*(D) = \operatorname*{argmin}_\beta \lambda ||\beta||^2 + \sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)). \quad (3)$$

By restricting the latent domains $Z(x_i)$ to a single choice, f_β becomes linear in β, and we obtain linear SVMs as a special case of latent SVMs. Latent SVMs are instances of the general class of energy-based models [18].

3.2. Semi-Convexity

Note that $f_\beta(x)$ as defined in (2) is a maximum of functions each of which is linear in β. Hence $f_\beta(x)$ is convex in β. This implies that the hinge loss $\max(0, 1 - y_i f_\beta(x_i))$ is convex in β when $y_i = -1$. That is, the loss function is convex in β for negative examples. We call this property of the loss function *semi-convexity*.

Consider an LSVM where the latent domains $Z(x_i)$ for the positive examples are restricted to a single choice. The loss due to each positive example is now convex. Combined with the semi-convexity property, (3) becomes convex in β.

If the labels for the positive examples are not fixed we can compute a local optimum of (3) using a coordinate descent algorithm:

1. Holding β fixed, optimize the latent values for the positive examples $z_i = \operatorname{argmax}_{z \in Z(x_i)} \beta \cdot \Phi(x, z)$.
2. Holding $\{z_i\}$ fixed for positive examples, optimize β by solving the convex problem defined above.

It can be shown that both steps always improve or maintain the value of the objective function in (3). If both steps maintain the value we have a strong local optimum of (3), in the sense that Step 1 searches over an exponentially large space of latent labels for positive examples while Step 2 simultaneously searches over weight vectors and an exponentially large space of latent labels for negative examples.

3.3. Data Mining Hard Negatives

In object detection the vast majority of training examples are negative. This makes it infeasible to consider all negative examples at a time. Instead, it is common to construct training data consisting of the positive instances and "hard negative" instances, where the hard negatives are data mined from the very large set of possible negative examples.

Here we describe a general method for data mining examples for SVMs and latent SVMs. The method iteratively solves subproblems using only hard instances. The innovation of our approach is a theoretical guarantee that it leads to the *exact* solution of the training problem defined using the complete training set. Our results require the use of a margin-sensitive definition of hard examples.

The results described here apply both to classical SVMs and to the problem defined by Step 2 of the coordinate descent algorithm for latent SVMs. We omit the proofs of the theorems due to lack of space. These results are related to working set methods [17].

We define the hard instances of D relative to β as,

$$M(\beta, D) = \{\langle x, y \rangle \in D \mid y f_\beta(x) \leq 1\}. \quad (4)$$

That is, $M(\beta, D)$ are training examples that are incorrectly classified or near the margin of the classifier defined by β. We can show that $\beta^*(D)$ only depends on hard instances.

Theorem 1. *Let C be a subset of the examples in D. If $M(\beta^*(D), D) \subseteq C$ then $\beta^*(C) = \beta^*(D)$.*

This implies that in principle we could train a model using a small set of examples. However, this set is defined in terms of the optimal model $\beta^*(D)$.

Given a fixed β we can use $M(\beta, D)$ to approximate $M(\beta^*(D), D)$. This suggests an iterative algorithm where we repeatedly compute a model from the hard instances defined by the model from the last iteration. This is further justified by the following fixed-point theorem.

Theorem 2. *If $\beta^*(M(\beta, D)) = \beta$ then $\beta = \beta^*(D)$.*

Let C be an initial "cache" of examples. In practice we can take the positive examples together with random negative examples. Consider the following iterative algorithm:

1. Let $\beta := \beta^*(C)$.
2. Shrink C by letting $C := M(\beta, C)$.
3. Grow C by adding examples from $M(\beta, D)$ up to a memory limit L.

Theorem 3. *If $|C| < L$ after each iteration of Step 2, the algorithm will converge to $\beta = \beta^*(D)$ in finite time.*

3.4. Implementation details

Many of the ideas discussed here are only approximately implemented in our current system. In practice, when training a latent SVM we iteratively apply classical SVM training to triples $\langle x_1, z_1, y_1 \rangle$, ..., $\langle x_n, z_n, y_n \rangle$ where z_i is selected to be the best scoring latent label for x_i under the

model trained in the previous iteration. Each of these triples leads to an example $\langle \Phi(x_i, z_i), y_i \rangle$ for training a linear classifier. This allows us to use a highly optimized SVM package (SVMLight [17]). On a single CPU, the entire training process takes 3 to 4 hours per object class in the PASCAL datasets, including initialization of the parts.

Root Filter Initialization: For each category, we automatically select the dimensions of the root filter by looking at statistics of the bounding boxes in the training data.[1] We train an initial root filter F_0 using an SVM with no latent variables. The positive examples are constructed from the unoccluded training examples (as labeled in the PASCAL data). These examples are anisotropically scaled to the size and aspect ratio of the filter. We use random subwindows from negative images to generate negative examples.

Root Filter Update: Given the initial root filter trained as above, for each bounding box in the training set we find the best-scoring placement for the filter that significantly overlaps with the bounding box. We do this using the original, un-scaled images. We retrain F_0 with the new positive set and the original random negative set, iterating twice.

Part Initialization: We employ a simple heuristic to initialize six parts from the root filter trained above. First, we select an area a such that $6a$ equals 80% of the area of the root filter. We greedily select the rectangular region of area a from the root filter that has the most positive energy. We zero out the weights in this region and repeat until six parts are selected. The part filters are initialized from the root filter values in the subwindow selected for the part, but filled in to handle the higher spatial resolution of the part. The initial deformation costs measure the squared norm of a displacement with $a_i = (0, 0)$ and $b_i = -(1, 1)$.

Model Update: To update a model we construct new training data triples. For each positive bounding box in the training data, we apply the existing detector at all positions and scales with at least a 50% overlap with the given bounding box. Among these we select the highest scoring placement as the positive example corresponding to this training bounding box (Figure 3). Negative examples are selected by finding high scoring detections in images not containing the target object. We add negative examples to a cache until we encounter file size limits. A new model is trained by running SVMLight on the positive and negative examples, each labeled with part placements. We update the model 10 times using the cache scheme described above. In each iteration we keep the hard instances from the previous cache and add as many new hard instances as possible within the memory limit. Toward the final iterations, we are able to include all hard instances, $M(\beta, D)$, in the cache.

[1] We picked a simple heuristic by cross-validating over 5 object classes. We set the model aspect to be the most common (mode) aspect in the data. We set the model size to be the largest size not larger than 80% of the data.

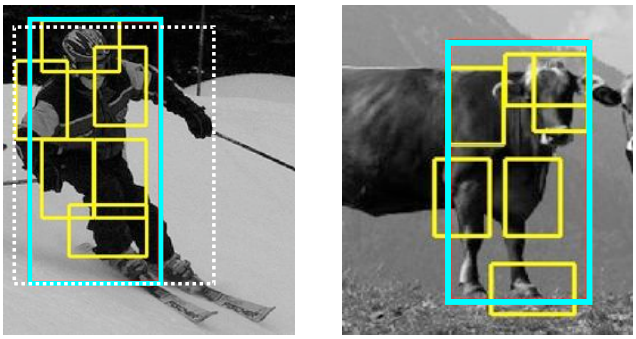

Figure 3. The image on the left shows the optimization of the latent variables for a positive example. The dotted box is the bounding box label provided in the PASCAL training set. The large solid box shows the placement of the detection window while the smaller solid boxes show the placements of the parts. The image on the right shows a hard-negative example.

4. Results

We evaluated our system using the PASCAL VOC 2006 and 2007 `comp3` challenge datasets and protocol. We refer to [7, 8] for details, but emphasize that both challenges are widely acknowledged as difficult testbeds for object detection. Each dataset contains several thousand images of real-world scenes. The datasets specify ground-truth bounding boxes for several object classes, and a detection is considered correct when it overlaps more than 50% with a ground-truth bounding box. One scores a system by the average precision (AP) of its precision-recall curve across a testset.

Recent work in pedestrian detection has tended to report detection rates versus false positives per window, measured with cropped positive examples and negative images without objects of interest. These scores are tied to the resolution of the scanning window search and ignore effects of non-maximum suppression, making it difficult to compare different systems. We believe the PASCAL scoring method gives a more reliable measure of performance.

The 2007 challenge has 20 object categories. We entered a preliminary version of our system in the official competition, and obtained the best score in 6 categories. Our current system obtains the highest score in 10 categories, and the second highest score in 6 categories. Table 1 summarizes the results.

Our system performs well on rigid objects such as cars and sofas as well as highly deformable objects such as persons and horses. We also note that our system is successful when given a large or small amount of training data. There are roughly 4700 positive training examples in the person category but only 250 in the sofa category. Figure 4 shows some of the models we learned. Figure 5 shows some example detections.

We evaluated different components of our system on the longer-established 2006 person dataset. The top AP score

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Our rank	3	1	2	1	1	2	2	4	1	1	1	4	2	2	1	1	2	1	4	1
Our score	.180	**.411**	.092	**.098**	**.249**	.349	.396	.110	**.155**	**.165**	**.110**	.062	.301	.337	**.267**	**.140**	.141	**.156**	.206	**.336**
Darmstadt							.301													
INRIA Normal	.092	.246	.012	.002	.068	.197	.265	.018	.097	.039	.017	.016	.225	.153	.121	.093	.002	.102	.157	.242
INRIA Plus	.136	.287	.041	.025	.077	.279	.294	.132	.106	.127	.067	.071	**.335**	.249	.092	.072	.011	.092	.242	.275
IRISA		.281					.318	.026	.097	.119			.289	.227	.221		**.175**			.253
MPI Center	.060	.110	.028	.031	.000	.164	.172	.208	.002	.044	.049	.141	.198	.170	.091	.004	.091	.034	.237	.051
MPI ESSOL	.152	.157	**.098**	.016	.001	.186	.120	**.240**	.007	.061	.098	**.162**	.034	.208	.117	.002	.046	.147	.110	.054
Oxford	**.262**	.409				**.393**	**.432**							**.375**					**.334**	
TKK	.186	.078	.043	.072	.002	.116	.184	.050	.028	.100	.086	.126	.186	.135	.061	.019	.036	.058	.067	.090

Table 1. PASCAL VOC 2007 results. Average precision scores of our system and other systems that entered the competition [7]. Empty boxes indicate that a method was not tested in the corresponding class. The best score in each class is shown in bold. Our current system ranks first in 10 out of 20 classes. A preliminary version of our system ranked first in 6 classes in the official competition.

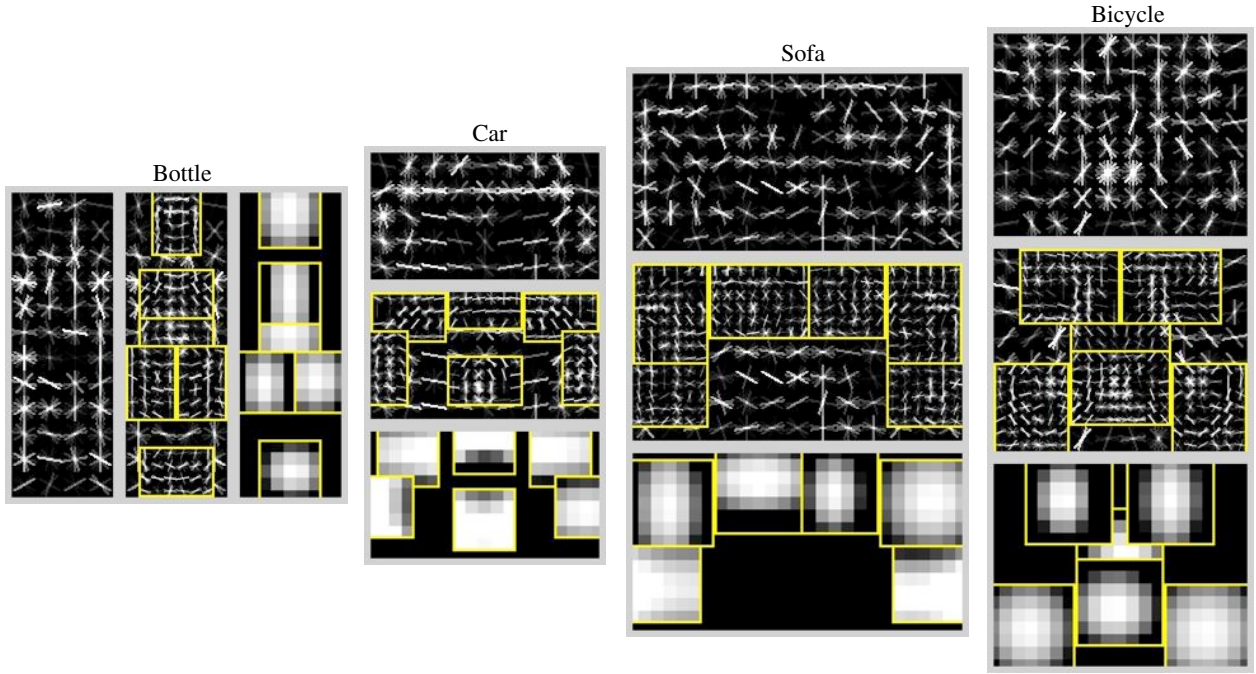

Figure 4. Some models learned from the PASCAL VOC 2007 dataset. We show the total energy in each orientation of the HOG cells in the root and part filters, with the part filters placed at the center of the allowable displacements. We also show the spatial model for each part, where bright values represent "cheap" placements, and dark values represent "expensive" placements.

in the PASCAL competition was .16, obtained using a rigid template model of HOG features [5]. The best previous result of .19 adds a segmentation-based verification step [20]. Figure 6 summarizes the performance of several models we trained. Our root-only model is equivalent to the model from [5] and it scores slightly higher at .18. Performance jumps to .24 when the model is trained with a LSVM that selects a latent position and scale for each positive example. This suggests LSVMs are useful even for rigid templates because they allow for self-adjustment of the detection window in the training examples. Adding deformable parts increases performance to .34 AP — a factor of two above the best previous score. Finally, we trained a model with parts but no root filter and obtained .29 AP. This illustrates the advantage of using a multiscale representation.

We also investigated the effect of the spatial model and allowable deformations on the 2006 person dataset. Recall that s_i is the allowable displacement of a part, measured in HOG cells. We trained a rigid model with high-resolution parts by setting s_i to 0. This model outperforms the root-only system by .27 to .24. If we increase the amount of allowable displacements without using a deformation cost, we start to approach a bag-of-features. Performance peaks at $s_i = 1$, suggesting it is useful to constrain the part displacements. The optimal strategy allows for larger displacements while using an explicit deformation cost. The follow-

Figure 5. Some results from the PASCAL 2007 dataset. Each row shows detections using a model for a specific class (Person, Bottle, Car, Sofa, Bicycle, Horse). The first three columns show correct detections while the last column shows false positives. Our system is able to detect objects over a wide range of scales (such as the cars) and poses (such as the horses). The system can also detect partially occluded objects such as a person behind a bush. Note how the false detections are often quite reasonable, for example detecting a bus with the car model, a bicycle sign with the bicycle model, or a dog with the horse model. In general the part filters represent meaningful object parts that are well localized in each detection such as the head in the person model.

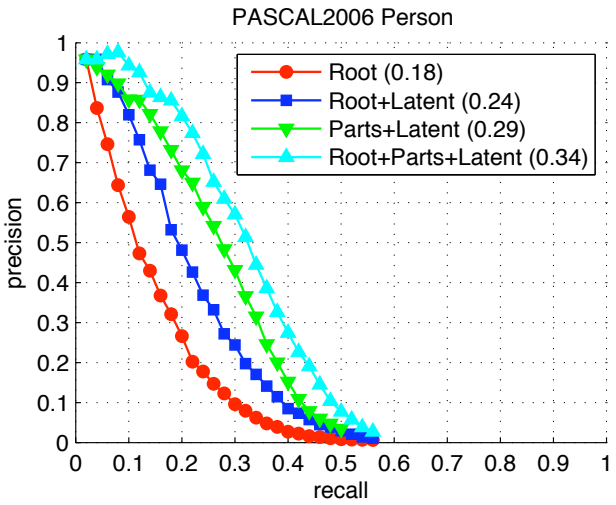

Figure 6. Evaluation of our system on the PASCAL VOC 2006 person dataset. *Root* uses only a root filter and no latent placement of the detection windows on positive examples. *Root+Latent* uses a root filter with latent placement of the detection windows. *Parts+Latent* is a part-based system with latent detection windows but no root filter. *Root+Parts+Latent* includes both root and part filters, and latent placement of the detection windows.

ing table shows AP as a function of freely allowable deformation in the first three columns. The last column gives the performance when using a quadratic deformation cost and an allowable displacement of 2 HOG cells.

s_i	0	1	2	3	2 + quadratic cost
AP	.27	.33	.31	.31	.34

5. Discussion

We introduced a general framework for training SVMs with latent structure. We used it to build a recognition system based on multiscale, deformable models. Experimental results on difficult benchmark data suggests our system is the current state-of-the-art in object detection.

LSVMs allow for exploration of additional latent structure for recognition. One can consider deeper part hierarchies (parts with parts), mixture models (frontal vs. side cars), and three-dimensional pose. We would like to train and detect multiple classes together using a shared vocabulary of parts (perhaps visual words). We also plan to use A* search [11] to efficiently search over latent parameters during detection.

References

[1] Y. Amit and A. Trouve. POP: Patchwork of parts models for object recognition. *IJCV*, 75(2):267–282, November 2007.

[2] M. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *ECCV*, pages II:628–641, 1998.

[3] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, pages 10–17, 2005.

[4] D. Crandall and D. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *ECCV*, pages I: 16–29, 2006.

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

[6] B. Epshtein and S. Ullman. Semantic hierarchies for recognizing objects and parts. In *CVPR*, 2007.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop.

[8] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[9] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Cornell Computing and Information Science Technical Report TR2004-1963, September 2004.

[10] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005.

[11] P. Felzenszwalb and D. McAllester. The generalized A* architecture. *JAIR*, 29:153–190, 2007.

[12] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[13] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, 22(1):67–92, January 1973.

[14] A. Holub and P. Perona. A discriminative framework for modelling object classes. In *CVPR*, pages I: 664–671, 2005.

[15] S. Ioffe and D. Forsyth. Probabilistic methods for finding people. *IJCV*, 43(1):45–68, June 2001.

[16] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, pages II: 2145–2152, 2006.

[17] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[18] Y. LeCun, S. Chopra, R. Hadsell, R. Marc'Aurelio, and F. Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[19] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *PAMI*, 29(10):1848–1852, October 2007.

[20] D. Ramanan. Using segmentation to verify object hypotheses. In *CVPR*, pages 1–8, 2007.

[21] D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *CVPR*, pages I: 206–213, 2006.

[22] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *IJCV*, 56(3):151–177, February 2004.

[23] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73(2):213–238, June 2007.