Modeling 3D viewpoint for part-based object recognition of rigid objects

Joshua Schwartz
Department of Computer Science
Cornell University
jdvs@cs.cornell.edu

Abstract

Part-based object models based on latent SVMs have proven to be a valuable tool for object recognition. We present a novel approach to learning object models with more sophisticated 3D structure. We develop a method for roughly inferring the viewpoint from which a photograph is taken. We then show how to use this information in the training and testing phases of learning a multi-component part-based model. We demonstrate state of the art results for multi-component models on the PASCAL Visual Object Categories 2007 dataset.

1 Introduction

Based on the results of the Pascal Visual Objects Challenge (a major international competition for building object recognition systems [2]), detection in certain classes is substantially easier than in others. All of the top methods in the VOC work well for certain object classes: cars, bicyles/motorcycles, and people, for instance. However, significantly worse results are seen in two types of categories: indoor furniture (sofa, chair, table) and non-human animals (cat, cow, horse). On these categories, accuracy is typically 50-70% worse than for cars or humans. Poor performance on non-human animals makes sense: animals articulate in complex ways and there may be substantial intra-class variation (a greyhound vs. a poodle vs. a St. Bernard, for instance). On the other hand, the results on indoor furniture are somewhat unexpected: why should all major detectors do 70% worse on, for instance, sofas than on cars?

This paper hopes to examine one possible reason: that categories like indoor furniture exhibit substantial variation in viewpoint (the position of the camera on the viewing sphere, as well as the distance to the object) compared with other categories. For example, figure 1 shows three examples of objects in the chair category in the Pascal VOC 2007 dataset. Visual inspection of the models learned by the leading algorithms shows that chair detectors typically place most of their energy on learning the "seat" and "leg" portions of the object by detecting horizontal planes and vertical bars. But, of course, planes with similar appearance can be found in a variety of non-chair objects (say, countertops) and vertical bars match well to many things (e.g. fences), hence the algorithms' high false-positive rates.

Detectors training on the easier categories typically identify some canonical parts and work by finding those parts: car/bicycle/motorcycle detectors, for instance, all gain substantial leverage by detecting wheels. Regardless of viewing angle, nearly all images of, say, a bicycle will have a wheel visible. In fact, Felzenszwalb et al showed that the model its method learns for cars even matches its "wheel" feature to headlights in frontal views of cars [3].

On the other hand, the harder categories may have less "canonical" parts across the whole category. Our hope is that by dividing into viewpoints, we separate out views that may actually have canonical parts that can be learned. For instance, the arrangement of four legs and a seat works well for most views of a chair, but not for views from above. Because most training procedures do not separate out

Figure 1: Three images of chairs from the Pascal 2007 dataset with vastly different appearances and viewpoints.

side views from top views, they must learn models that simultaneously match both types of views (and, hence, do not match either type as well as we might hope).

This work is a direct extension of the deformable part model of [3]. We choose this model to modify for several reasons: it is a state of the art model, winning the Pascal VOC for several years as well as the "lifetime achievement award" for this contest; many subsequent methods have been extensions to this model; the basic code is available for download [5] ; and, it presents an interesting machine learning algorithm (the latent SVM) that relates nicely to our course material and provides a good environment for adding viewpoint, with several challenges that we solve herein in the learning procedure. We use this formulation, but train detectors based on viewpoint information that we infer separately. The work of Su et. al [10] also incorporates viewpoint into object detection; by comparison, our model is substantially simpler, and it requires less structured training data.

The main contributions of this paper are:

- We provide a method for inferring camera viewpoint of an object. In order to train our system to use this method, we create a new dataset of 3D object models by extending prior research on 3D reconstruction for outdoor scenes.
- We show how this orientation estimator is useful in initialization for the Latent SVM formulation of [3]. The authors of that paper note that their training method is very sensitive to initialization; we demonstrate that sensitivity and show that using our viewpoint estimates substantially improve upon the previous solution.
- We show how this orientation estimator can also be incorporated into the set of features used in the lSVM classifier.
- Of note: our model gives not just a detection, but an orientation estimate for objects detected.

We apply our method to sofa and chair categories in the Pascal VOC dataset, though it could just as well be applied to others; I have chosen these categories for simplicity and because most existing methods do so poorly on them.

2 Pictorial Structures Models

We use a pictorial structures model for objects categories: a given category has a root filter that determines the location of the object centroid and a set of part filters that look for individual object parts. Each part filter encodes a distribution over locations as offsets from the center of the root filter. Given a trained model, efficient algorithms exist for finding the best match of the model to an image. Visualizations of root, part, and part location models can be seen below in figure 4.

Following [3], we use the Histogram of Oriented Gradients (HOG) feature to encode appearance. This feature simply encodes a histogram of gradient orientations at various scales, giving a 36-dimensional feature vector. Since we use exactly the same appearance and part geometry model as [3], and since this project is focused on machine learning and not vision problems, I do not give a full description of the HOG filter and the general framework of pictorial structures. [3] gives a substantial discussion of these methods.

Inputs: set of positive examples P and negative examples N
Initialize $\beta_1 ... \beta_M$ [1]
for $t = 1...T$ **do**
  // Do hard partition of data based on current values for $\beta_1 ... \beta_M$
  $P_1 = P_2 = ... = P_M = \emptyset$
  for $I \in P$ **do**
    $m = \max_{m=1...M} \beta_m \cdot \phi(I)$
    $P_m = P_m \cup I$
  end for
  // Now train each β_m using only those positive examples assigned to it
  for m = 1...M **do**
    $\beta_m = \text{TrainSVM}(P_m, N)$
  end for
end for
Output: $\beta_1 ... \beta_M$

Figure 2: Algorithm for training a latent SVM, where TrainSVM(P, N) is the standard algorithm for training an SVM given positive examples P and negatives N.

Critically, learning a pictorial structures model can be encoded as learning an SVM, where the feature space encodes appearance and part geometry.

3 Learning

Our data consists of a set of positive training examples (images along with bounding boxes denoting instances of a particular object class in that image) and negative training examples (images that do not contain the class we are trying to learn). For each object class, the goal is to learn a set of SVMs encoding pictorial structures, one SMV for each view from a discrete set of possible viewpoints. We term these SVMs *components* of our model for an object category. Each SVM discriminates between that viewpoint and background. However, our training data does not come with viewpoint labels, so viewpoint must be treated as a hidden variable. We use the latent SVM framework to deal with the hidden information.

3.1 Latent SVM

Suppose we have some feature mapping from images to real-valued feature space: $\phi : I \rightarrow \mathbb{R}^d$ and sets of positive and negative training images P and N. A standard SVM finds a weight vector $\beta \in \mathbb{R}^d$ that maximizes the linear margin between the set of positive and negative examples in feature space, and for a training example assigns a score:

$$S_{SVM}(I) = \beta \cdot \phi(I)$$

In a latent SVM (lSVM), we assume that each object instance was generated from one of M components of our object category. For instance, one component might correspond to frontal views and one to side views. If we have M models $\beta_1 ... \beta_M$, then the score of an image is:

$$S_{lSVM}(I) = \max_{i \in 1...M} \beta_i \cdot \phi(I)$$

The intuition behind using a $\max$ rather than, for instance, a sum is that we want to give a positive detection only if an image matches some model component well.

In training, however, the assignment of positive examples to particular components is not given, so it has to be treated as latent information. To make the learning procedure tractable, an iterative approach is used. At each iteration, the positive training data is partitioned into sets of images that match different components best; components are then trained just using those images as positives. The full algorithm for training is given in figure 2.

Now, we show how to incorporate viewpoint information into the above formulation by making components correspond to various viewpoints.

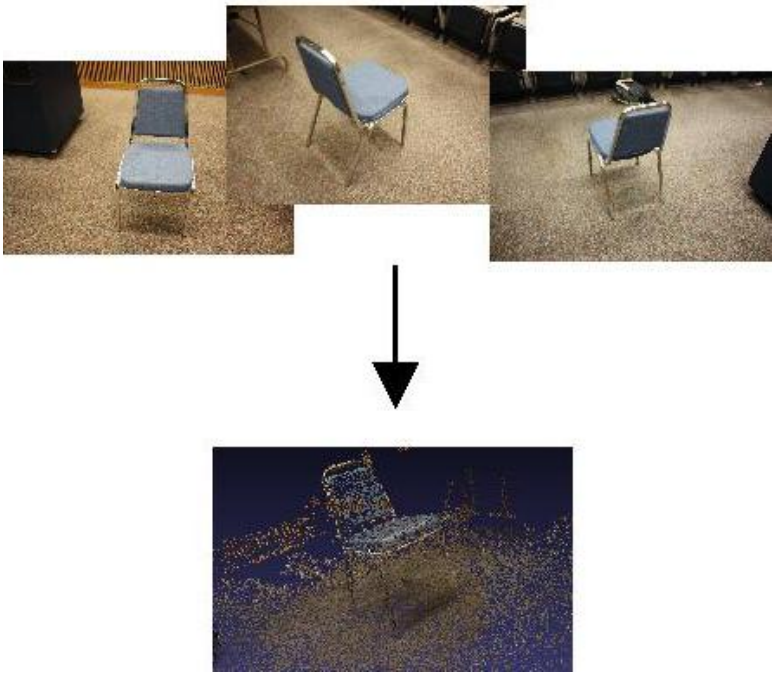

Figure 3: A collection of images of a chair is turned into a 3D point cloud. This process is repeated for several objects to generate a set of 3D clouds.

4 Viewpoint Estimation

Since viewpoint is a hidden variable in our training data, we bootstrap the learning process by inferring viewpoint using a regression model we learn from more structured data.

4.1 Building 3D models

We create a sparse 3D model of a few specific examples of an object class by using the Bundler method of Agarwal et al. [1]. This method, originally intended for use in outdoor scenes, creates a 3D point cloud representing an object, given a set of images of that object. It matches SIFT features across images and uses those correspondences to infer camera parameters. We then use the multiview stereo method of Furukawa et al. [6] to obtain a dense model. An example of several images and one view of the dense model created is shown in Figure 3. To align these models and put them in the same coordinate system, we then detect the groundplane using a Hough transform, align the ground planes, align the object centroids, and finally search over 1D rotations to find the best alignment. We are then able to easily detect the mass of the chair in the 3D space and project that back into each 2D image to obtain an estimate of the bounding box.

This gives us a set of 3D models that are registered to one another. More significantly, it gives us an accurate estimate of the camera position with respect to the model, for each training image, as well as a bounding box around the object in each image.

We manually created twelve 3D object models by taking roughly 40 photographs each for a set of 12 chairs. The end result of this procedure is that we have 480 images of chair, each with a bounding box around the chair and a fairly reliable estimate of viewpoint.

4.2 Training and Inference

We learn a multinomial regression model for estimating viewpoint, using the data generated through the process described in Section 4.1. For each individual image associated with a 3D model, we apply the HOG filter to the bounded region. This produces a real-valued feature vector. The process

	Random partition, 4 comp.	[3]	Our results, 4 comp.	6 comp	8 comp
Chair	0.161	0.163	0.187	0.206	**0.209**
Sofa	n/a	0.216	**0.276**	0.261	0.245

Table 1: Average precision. Comparison of our results to those from the original method of [3]. In the random initialization, we simply assign images to clusters randomly; the figure reported is the average accuracy over 3 runs. In [3], training images are assigned to clusters using the aspect ratio of their bounding boxes. In our results, we use our viewpoint estimator in initialization and as an additional set of features. Note that the chair category has roughly 4x the amount of training data as the sofa category.

in Section 4.1 also generates viewpoint labels for each image. We bin the set of views into K bins, where K is the desired number of components in our model, and treat bin membership as drawn from a multinomial distribution. We learn weight vectors $w_1 ... w_K$ such for a given feature vector ϕ, the probability of view v_k is:

$$\Pr[v_k|\phi] = \frac{\exp(\phi \cdot w_k)}{1 + \sum_{k=1}^{K-1} \exp(\phi \cdot w_k)}$$

for $k = 1...K-1$ and

$$\Pr[v_K|\phi] = \frac{1}{1 + \sum_{k=1}^{K-1} \exp(\phi \cdot w_k)}$$

In practice, to learn these parameters w_k we use the Matlab `mnrfit` function. This gives us a set of weights we can use to predict viewpoint of a new section of an image.

5 Incorporating viewpoint into object models

As described in figure 2, at each stage of the lSVM training, we assign each training example to the model for which it receives the highest score; then, we update each model using only the training examples assigned to it at that stage; we repeat until convergence.

Of course, this method must be initialized: in the case of [3], the authors simply order the bounding boxes of training examples by aspect ratio,[2] divide that ordered list of bounding boxes evenly into chunks and train one SVM component per chunk. The authors' idea of dividing objects into bins based on aspect ratio is that aspect ratio variation often corresponds to viewpoint variation (e.g. in the case of cars). However, for chairs for instance, changes in viewpoint do not substantially change aspect ratio. In practice, this method is very sensitive to initialization. We demonstrate this quantitatively in our results.

Rather than binning by aspect ratio, we instead apply the model learned in Section 4.2, labeling each training example with its most likely viewpoint. We then train one SVM for each viewpoint based on the images that receive that label, and use this as our initialization.

Additionally, for each bounding box, we compute the probability it was drawn from each of the K viewpoints under our model, and add this probability to the feature vector for this box. Hence, the viewpoint estimates are incorporated into the training and testing phases.

6 Results

We demonstrate results on the Pascal VOC 2007 dataset. The 2007 "chair" set contains 224 positive training images with 400 total chair objects and 221 validation images with 398 chair objects. The 2007 "sofa" set contains 111 training images with 124 sofas, 118 validation images with 124 sofas, and 229 test images with 248 sofas. A large set of negative training and test images is also specified. All positive images come with human-labelled bounding boxes denoting the location of objects.

[2]The training data for the Pascal VOC consists of images with all positive examples denoted by human-labelled bounding boxes.

In testing, a successful match is recorded if its bounding box has at least 50% overlap with the human-labelled bounding box.

We learn object class models with 4, 6, and 8 components. Figure 4 shows the 4 component model learned by our method. Each row is one viewpoint model. The left column shows root filters, the middle column shows part filters, and the right column shows part locations. Visual inspection of the root filters gives some insight into what views of chairs are matching what model: clearly the first and second rows correspond to angled views of chairs and the third and fourth rows correspond to front and back views.

Table 1 shows average precision for 4, 6, and 8 component models learned by our system (that is, 4, 6, and 8 discrete viewpoints)as well as the results reported in [3][3] and results for a 4 component model initialized randomly rather than using viewpoint estimation. Results for the chair model continue to improve as additional components are added. However, results for sofas are best for the 4 component model and decline rapidly with more components. This makes sense: since there are only roughly 120 training examples for sofas, when the training examples are partitioned into more and more sets, the number of examples corresponding to any given component is very small, so the model is likely to overfit to the data. Figure 5 shows precision/recall curves for four component models on chairs and sofas.

Our results on these categories are dramatically better than those reported in [3], which is the original formulation that we modified to encode viewpoint.[4] For four component models, we improve average precision by over 35% for sofas and almost 9% for chairs; our 8 component model improves average precision by roughly 15% for sofas and 25% for chairs.[5] Certainly there is still a large margin for improvement, but the results here are closer to in line with those that [3] gets on other categories in the Pascal dataset, so it appears that incorporating viewpoint begins to close the gap between "'easy" categories and harder ones.

7 Extensions

We have demonstrated that incorporating viewpoint into the training process shows promise for detecting indoor object categories. There are several clear lines of research that might result in better detection performance:

- Improved viewpoint estimators. While our viewpoint estimator appears to do a good job of clustering views into useful categories, a more sophisticated detector could provide better results. A detector that did not require highly structured data for training would also be preferable.
- More sophisticated learning procedure. The lSVM learning procedure makes a hard assignment of objects to components for each stage of training. While this makes learning convex, the tradeoff is that the learning may be less accurate. This dichotomy is exactly the one between hard-EM and regular EM, and EM is often preferable, so there is reason to think that it would be useful here.
- Training with larger datasets. Our preliminary results suggest that as the amount of training data goes up larger number of components can be learned, and that progressively better results can be achieved. If we had a very large dataset, it would be interesting to examine how results improve as we scale the number of components for a given object category.

I plan to address these issues in future research.

[3] We report their "base model" results. They are able to achieve a slightly higher accuracy (still well below ours) by using context to rescore matches. The context rescoring could just as easily be applied to improve our method, but is beyond the scope of this project, so those results are omitted here.

[4] Note also that [3] was the winning entry in the 2007 Pascal VOC.

[5] By "improves," I mean the percentage by which accuracy improved. So, for instance, since [3] reports an accuracy of 0.163 for chairs and our 4 component model gets an accuracy of 0.187, our reported improvement is $(0.024/0.163 = 14.72\%)$.

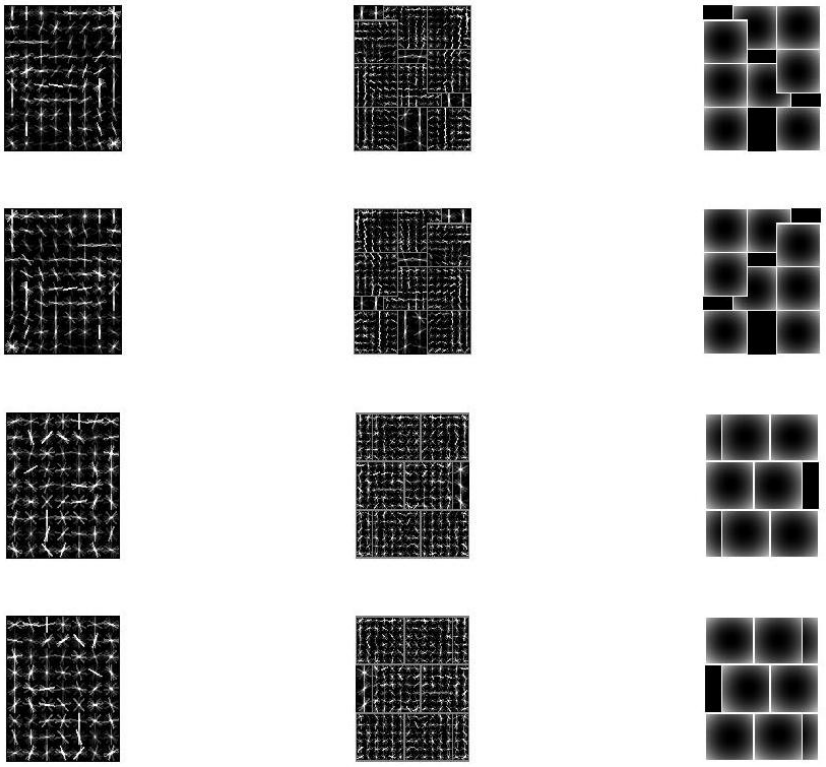

Figure 4: 4 component chair model learned by our algorithm. The left column is the root filter for each component. The middle column shows the part filters. The right column shows the likelihood of part locations.

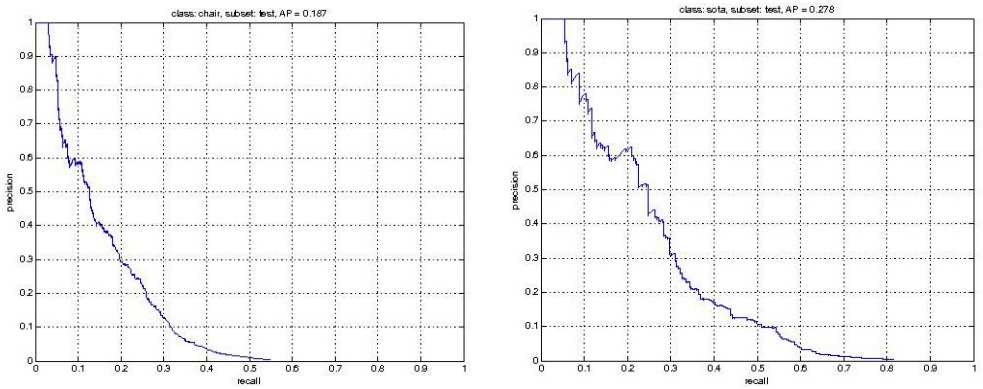

Figure 5: Precision recall curves for our method, 4 component model. Results on the chair category are on the left and sofa category are on the right.

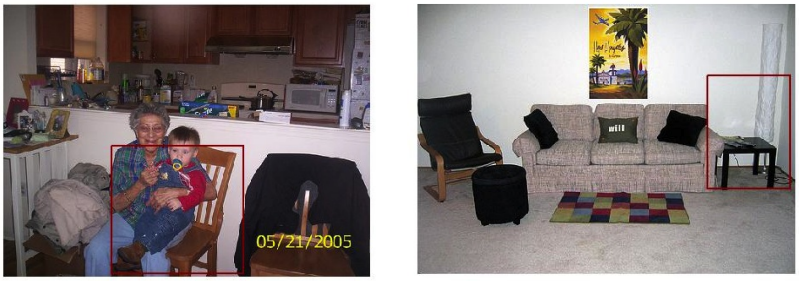

Figure 6: Example detections for the four component chair model. The left image is a true detection (despite heavy occlusion) and the right image is a false positive.

8 Acknowledgements

This is joint work with Dan Huttenlocher and Noah Snavely.

References

[1] S. Agarwal, N. Snavely, S. Seitz, and R. Szeliski. Bundle adjustment in the large. In *ECCV*, 2010.

[2] 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/index.html.

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

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

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

[6] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, 2007.

[7] Y. Li, D. Huttenlocher, and N. Snavely. Location recognition using prioritized feature matching. In *CVPR*, 2010.

[8] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *CVPR*, 2003.

[9] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

[10] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV*, 2009.