Object Detection Using a Deformable Parts Model

Evan Rosen and Nikil Viswanathan
Computer Science Department
Stanford University
{emrosen,nikil}@stanford.edu

1. Introduction

We implemented the deformable parts model described in [3]. We first give a brief high-level description of the deformable in part model and the histogram of gradients (HoG) features including some of the key insights upon which they rely. In Section 3.1 we give a detailed description of the model in [3]. In Section 3.2 we note some of the ways in which our model differs from that presented in the original paper. Next, we give a detailed description of our code in Section 4. We then present our results and evaluate the trade-offs of various design decisions in Section 5. Finally, we discuss some of the implementation constraints we encountered during for this project and give suggestions for improvement in Section 6.

2. Person Detection

The problem of object detection requires finding representations for objects which will remain invariant under intra-class variation. For many objects, uniformities with respect to shape texture and color can yield robust detection algorithms. Unfortunately in people these features are rarely invariant across instances due to changes in clothing, scene context and articulation. This rules out some of the more efficient object detection techniques which can efficiently compute filter cross-correlations using optimizations like the integral image representation of Viola and Jones [6]. Despite the challenges posed by articulation, shape remains one of the more important visual features for person detection. It is not the blue of a sweater that reliably identifies a person, but the shape of a head or the thickness of an arm. Moreover, modeling the space of possible articulations is an especially difficult problem in part due to the curse of dimensionality. The relative independence of human points of articulation means that people can take on a combinatorially large set of configurations. Learning these configurations directly would then require an enormous amount of training data.

A recent approach to dealing with the high-dimensionality of people configurations has been to decompose people (and objects in general) into a set of independent parts. For example, though articulation of the neck allows the shape of the torso and head combined to change, it leaves the shape of each part relatively unchanged. The Implicit Shape Model [4], the deformable parts model [3] and even visual words models such as [2], all take advantage of the independencies in the data by learning part-level models which remain relatively invariant across instances.

While decomposing shape alone has proven useful, the assumption of complete independence between parts is clearly too strong. Ideally we would have a set of models for different features of an object such that each models complexity would be proportional to the invariance of that feature. This way, given a fixed training size, those models for highly variable features would not overfit, but those models for more invariant features would take full advantage of the signal. The systems of [4] and [3] take an approach similar to this by modeling invariant patches or keypoints in relatively great detail, while modeling the configuration of those keypoints more coarsely.

3. Algorithm

3.1. Deformable Parts Model

The deformable part model in [3] implements the intuitions discussed above by representing a person as a set of part appearance models whose locations are modeled by a star-shaped configuration model. In addition to the part models, a single root root is used to represent the appearance of the entire person, despite the changes in configuration between instances. Each of these appearance models uses the Histogram of Gradients (HoG) features to represent the corresponding patch in the image. The entire model is then formulated as a binary classification problem in which we wish to learn the characteristics of the root and part feature patches as well as their spatial configuration. Specifically, the model consists of a set of SVM weights which correspond to the HoG feature weights for the root and part filters plus a set of linear and quadratic deformation penalties by which encode the spatial configuration of the parts with respect to the root filter.

A key part of their approach involves the ability to simultaneously learn the appearance and configuration models while also estimating the location of each part for each instance. Without this additional step, it is not possible to use the notion of independence, discussed above to overcome the articulation problem. [3] use an algorithm in the style of EM to iteratively learn the optimal appearance and configuration models for some placement of parts in each image and then find the optimal part placements given the updated appearance and configuration models. Lastly, they employ a principled technique for finding hard negatives, by updating their negative training examples after each iteration with the labelled negative examples that are the most misclassified.

3.2. Our Model

Our model differs from [3] primarily in the way we do non-maximum suppression in addition to several other small aspects. For non-maximum suppression, we take a set of detections and order by confidence. Proceeding down this list in decreasing confidence, we select the current detection and then remove any other bounding boxes which overlap with the current bounding box by more than 25

We also experimented with automatic root filter size selection, though we eventually converged onto the same parameters used in [3]. We first plotted the x and y dimensions to see the distributions of scales in Figure 1(b). This skewed nature of this distribution suggested that an arithmetic mean would be biased by the the larger outliers and be a poor fit for most bounding boxes. We realized however, that the scale was only on part of the root filter dimensions. Due to the anisotropic scaling preformed as a pre-processing step, it is also important to choose an aspect ratio which does not require too much distortion of the original bounding box. The distribution of aspect ratios (y/x) is plotted in Figure 1(a) which suggests that a value between one and two would be optimal. Using the median value for each dimension yielded a root filter dimension of 229×125 which has an aspect ratio of 1.8 and falls in a densely populated part of the scatter plot. Despite our principled approach, we found that setting the dimensions to 128×64, the size used in the paper, actually improved our results significantly. While we cannot be sure how much of this has to do with overfitting, the remainder of results reported will assume the 124×64 root filter dimensions.

4. Code

The main file of our code is run_tests.m. run_tests has the parameters and test framework. To run the entire program, call run_tests.

Key Files:

- **run_tests.m**:
  Run this to start the entire pipeline.
- **train_detector.m**:
  The pipeline for training the detector including the latent SVM loops and the hard negative mining loops.
- **get_pos.m**:
  Extracts bounding box statistics, rescales positive training examples and computes HoG features.
- **get_neg_rand.m**:
  Extracts random negative training examples such that they do not overlap with positive examples and computes HoG features.
- **train_detector.m**:
  The pipeline for training the detector including the latent SVM loops and the hard negative mining loops.
- **test.m**:
  The framework for evaluating the detector.
- **detect.m**:
  The actual detection sliding window code for a given image.

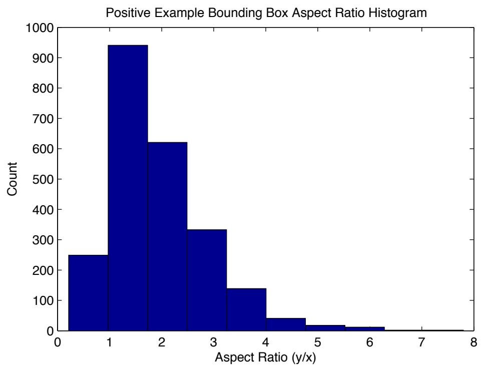
Several files have a debug parameter at the top; setting this parameter to true prints debug statements and displays visualizations. We wrote all of our code in our system with the exception of the HoG feature generation, the SVM solver, and part of the root filter visualization. Due to computational constraints we chose to use the optimized C++ HoG feature implementation from [3]. While implementing this ourselves would have been an informative process, having recently implement SIFT for 223B, we felt comfortable using the code without actually writing it ourselves. Computational efficiency and an interest in the vision components of the algorithm also led us to use the liblinear SVM solver [1] as a component of our latent SVM implementation.

5. Results

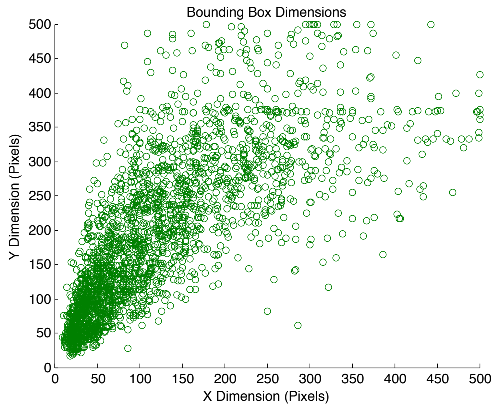
The best results we achieved were on a run which used the entire training data set (2358 positive, 2498 negative) testing on 100 examples. The precision and recall curve for this run is shown in Figure 2. Despite the fact that our system never reached competitive levels of performance, we were able to evaluate a variety of design decisions through their relative impact on the performance of the model.

5.1. Upsampling vs Downsampling

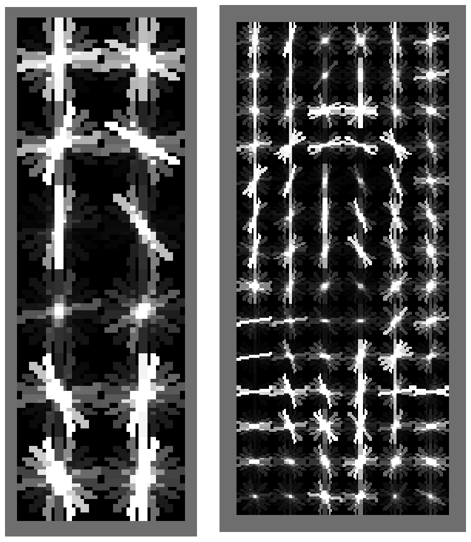
Given the constraint that the parts filters be computed at twice the resolution of the root filter, we either needed to use a downsampled version of the original bounding box for the root filter HoG features or an upsampled version for parts filters. While this decision has no effect on the testing method, it does bear upon the resolution of the part filters and the general complexity of the model. We reasoned that if we downsampled the original image to get the root filter



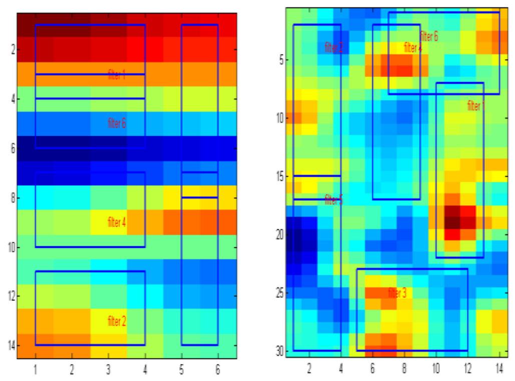
(a) this is a test caption



(b) scatter plot of positive bounding box dimensions



(c) Downscaled (d) Upscaled



(e) Downscaled (f) Upscaled

Figure 1. Subfigures (d) and (c) show the positive weight HOG feature visualization for the initial root filters. Subfigures (f) and (e) show a heat map of the initial root filters with the initial part locations drawn on MIT Data Set

resolution we would be throwing away valuable information for training the root filter. On the other hand, upsampling the images for parts filters cannot create new information with which to train the part filters, but merely trains a more complex model on the same data.

We initially downsampled our root filters on the thinking that they need only pick out coarse patterns of person like shapes. However, when training our initial root filters on the MIT pedestrian data set [5], we noticed that the root filter did not look at all like a person and the parts did not have any correspondence to human body parts as shown in Figures 1(c) and 1(e) in comparison to upsampling as shown in Figures 1(d) 1(f). We werent sure to what degree this was just a result of the resolution at which we can pick out human shapes in HoG feature visualizations or whether it really indicates that our model suffered from using too course of HoG cells, which lost valuable spatial information. However, looking at the parts initialization, we noticed that the parts in the upsampled image corresponded with the physical model of individual parts in a human whereas the downsampled image seemed to have non-corresponding parts initialization.

5.2. Number of Parts

Given that our the parts models do exhibit an especially intuitive configuration, we thought it might be useful to try varying the number of parts in the model . Keeping the total proportion of the root filter covered by the parts fixed at 0.8, we experimented with 2,4,6, and 8 parts. The results are shown in Figure 3. Unfortunately this presented no clear winner, potentially due to the small train (200 positive and 500 negative) and test (200) data sets.

5.3. Root Filter Evolution in Latent SVM

Our main goal of using the MIT pedestrian data set was to provide a sanity check that we were correctly creating the feature matrix correctly and implementing the latent SVM training. The HoG visualizations of our root filter weights

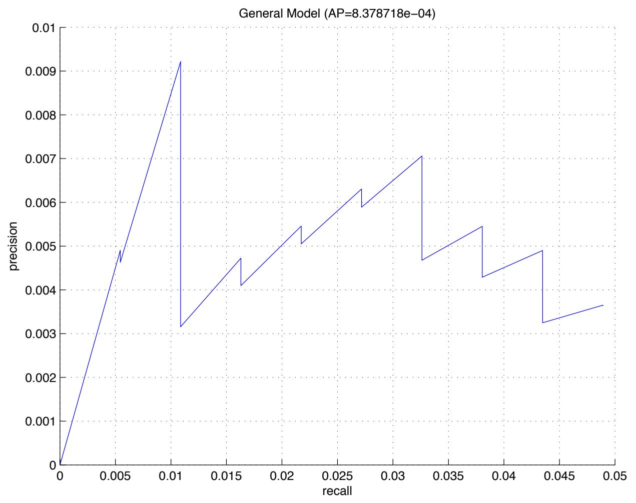


Figure 2. Precision recall curve for model trained on entire data set

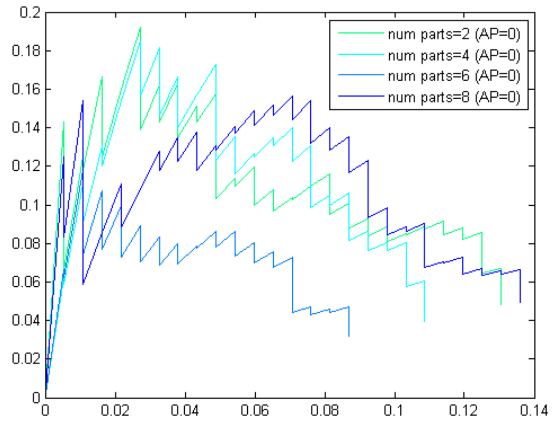


Figure 3. Precision Recall curves for varying number of parts models while keeping the proportion of the root filter covered by parts model fixed at 0.8

in the first row of Figure 4 show the initial values produced by training without the parts models, and the incremental results from the first two epochs of latent SVM for the MIT data set. These show a clearly pattern of a human form and convinced us that we were on the right path at least. The changes between iterations are both interesting and hard to interpret due to intensity normalization between each filter visualization (the intensities in each filter are normalized by the maximum absolute feature value). Setting aside the intensity, we can at least see that the shape of the human is becoming clearer as the parts model locations begin settle and become more discriminative, letting the root filter focus more on invariant body shape features.

On the other hand, our results on the PASCAL VOC data

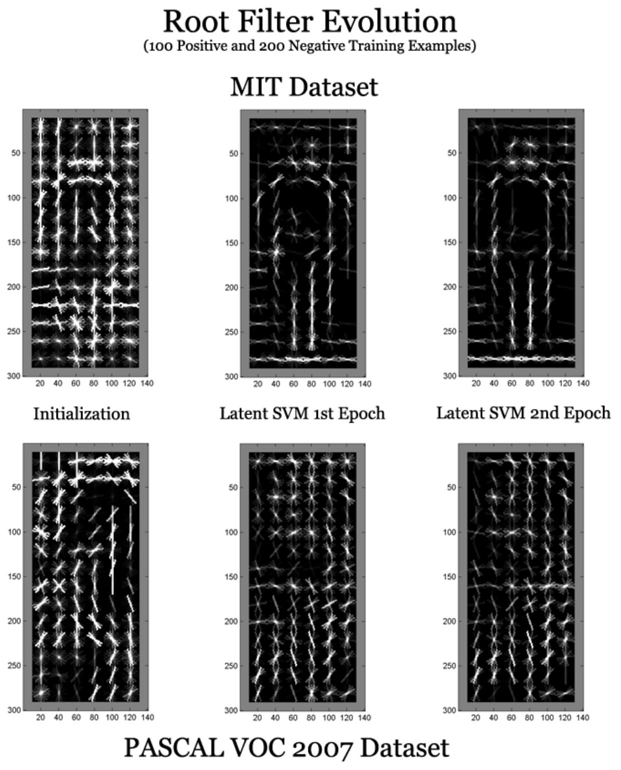


Figure 4. Root filter evolution on MIT and PASCAL VOC 2007

display no clear human like form. Looking at the actual images this understandable because many of the variety of poses is great and our training set size is quite small (100 positive). Though the progression of the root filter on the PASCAL data set is less human like, the same effect of increasing feature sparsity can be observed.

5.4. Mining Hard Negatives

Unfortunately we were not able to generate large scale results to test the efficacy of this component. However, it is worth noting that when searching for new hard negatives on the first iteration, almost every sampled image patch turned out to be a hard negative. This points to severe overfitting which we expect the hard negatives to help combat.

5.5. Detection Window Size Bias

One interesting pathology of our model is the tendency to make the most confident guess at the largest level of the spatial pyramid (meaning the smallest detection windows). However, if we restrict the levels of the spatial pyramid searched at test time, we see that the system is still predicting larger bounding boxes, but they simply do not making it through non-maximum suppression in the presence of smaller detection. We should expect to see at least a few larger windows which correspond to the most confident guesses. Recall, that our greedy non-maximum suppression

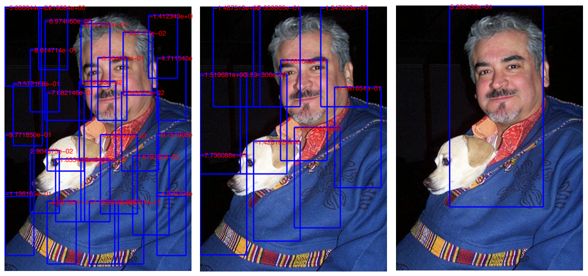


Figure 5. Examples of size bias in detection windows where detection windows are limited to the highest 10, 5, and 3 levels of the pyramid

algorithm never removes an overlapping detection window if it is the more confident of the two windows. Another possibility is that the sheer number of candidate detection windows is much greater at the lower levels of the spatial pyramid (larger images). Because we use a sliding window with a fixed step size in cell space, we wind up covering the entire image in only two or three steps at the higher levels of the spatial pyramid (smaller images) and generated proportionately fewer guesses. Two examples of this behavior are shown in Figure 5 which consider the top 10, 5 and 3 levels of the spatial pyramid.

5.6. Effect of Training Set Size on Root Filter

As we were not able to access machines to run the entire training set we attempted to at least measure the effect of training set size on the meager data which we could manage, the hopes that we might extrapolate on our models performance were we to use the entire data set. Figure 6 shows the root filters trained on data sets of size 100 200 and 300. We can see that there is quite a bit of improvement in terms of sparsity as the number of examples grows.

5.7. Effect of Training Set Size on Precision and Recall

Increasing the training set size from 50 to 100 to 200 had a drastic improvement in terms of the precision and recall for our model. The precision of the 200 example trained classifier was better over the entire recall curve than any point on the 100 example trained curve. We believe that this indicates our model and algorithms are at least somewhat correct and that if we were able to run the it on the full data set we would see even more drastic performance improvements all around.

6. Implementation Constraints

We ran into several computational issues while testing and running our code. First of all, our AFS drives did not have enough space to store all of the image files from the

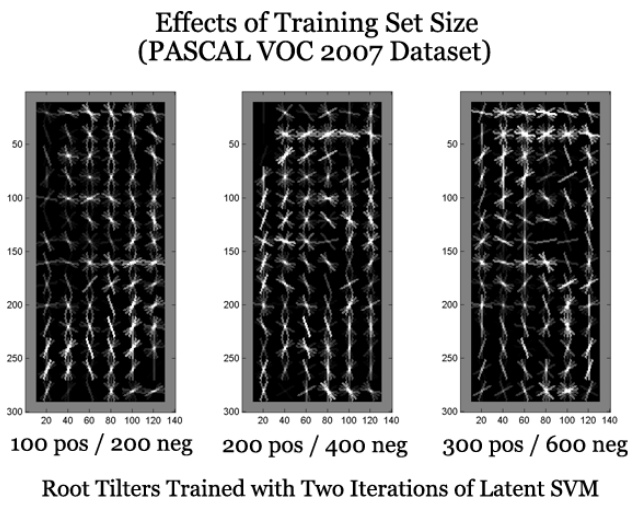

Effects of Training Set Size
(PASCAL VOC 2007 Dataset)
50
100
150
200
250
300
20 40 60 80 100 120 140
100 pos / 200 neg
200 pos / 400 neg
300 pos / 600 neg
Root Tilters Trained with Two Iterations of Latent SVM

Figure 6. Effect of training set size on root filter

PASCAL VOC data set and we struggled with various issues relating to this for over a week. We then ran the code on our personal machines which were laptops with 2GB of RAM however as we tried to train on a data set of over 400 positive and 400 negative images we found our machines crashing on the SVM training method. When running our classifier on test data, each image took many seconds to classify so when we were trying to rapidly run tests for tuning parameters we would cut down the test set to around 200 examples, or slightly more if we wanted good graphs and visualizations.

References

[1] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[2] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision*, 2004.

[3] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. Citeseer, 2008.

[4] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 17–32. Citeseer, 2004.

[5] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. pages 193–99, 1997.

[6] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. 2001.