Using Segmentation to Verify Object Hypotheses

Deva Ramanan
Toyota Technological Institute at Chicago
Chicago, IL 60637
ramanan@tti-c.org

Abstract

We present an approach for object recognition that combines detection and segmentation within a efficient hypothesize/test framework. Scanning-window template classifiers are the current state-of-the-art for many object classes such as faces, cars, and pedestrians. Such approaches, though quite successful, can be hindered by their lack of explicit encoding of object shape/structure – one might, for example, find faces in trees.

We adopt the following strategy; we first use these systems as attention mechanisms, generating many possible object locations by tuning them for low missed-detections and high false-positives. At each hypothesized detection, we compute a local figure-ground segmentation using a window of slightly larger extent than that used by the classifier. This segmentation task is guided by top-down knowledge. We learn offline from training data those segmentations that are consistent with true positives. We then prune away those hypotheses with bad segmentations. We show this strategy leads to significant improvements (10-20%) over established approaches such as ViolaJones and DalalTriggs on a variety of benchmark datasets including the PASCAL challenge, LabelMe, and the INRIAPerson dataset.

1. Introduction

One of the open issues in object recognition is the role of segmentation. Several issues remain unclear. Can one *quantitatively* demonstrate that segmentation improves detection performance? If so, how does one computationally detect/segment in an efficient manner?

We address these issues with a simple but surprisingly effective *hypothesize-and-test* framework. We leverage the successful work on sliding-window pattern-recognition detectors. We use these as *attention* mechanisms that propose many hundreds of object hypotheses per image. By computing a local **figure-ground segmentation** at hypothesized detections, we show one can prune away many false hypotheses. We quantitatively show this strategy significantly boosts the performance of the baseline sliding-windows detectors.

Figure 1. *Window-based classifiers are the state-of-the-art for object detection across many categories. These approaches typically compute some linear function of edge-like features (such as thresholded Haar wavelets or oriented gradients). Such approaches, though quite successful, can suffer from a lack of explicit encoding of object structure. We show typical false positives above. On the* **left***, the face detector becomes confused by edges in foliage. The pedestrian detector (***top***) mistakens strong vertical edges for a person, while the car detector (***right***) likes to fire on strong horizontal edges. We propose to use figure-ground segmentation cues in conjunction with edge-based window classifiers. For example, one can remove the last false-positive person by explicitly reasoning about what pixels belong to the object versus the background.*

In this work, we use well-known state-of-the-art baselines – ViolaJones [16] for finding faces and DalalTriggs [3] for finding pedestrians and cars. Such window-based classifiers perform quite well in practice - the DalalTriggs detector yields the top score across many object classes (including people and cars) from the PASCAL 2006 Visual Object Challenge [4]. However, such "pattern-recognition" approaches could be limited by their lack of encoding of object shape/structure - one might find faces in trees or mistake pillars for pedestrians (Fig. 1).

To address these limitations, we use a *verification* stage that computes a local figure-ground segmentation around the candidate detection window. Though image segmen-

1-4244-1180-7/07/$25.00 ©2007 IEEE

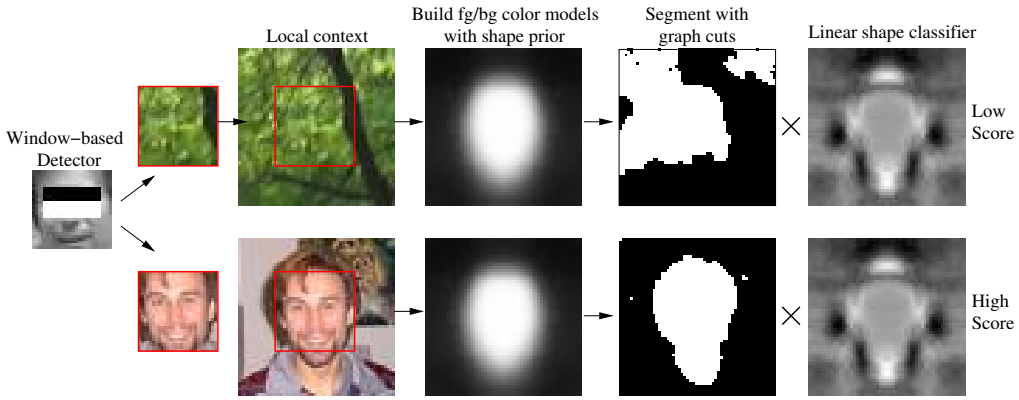

Figure 2. *Our algorithm for combined detection and segmentation. Given a putative detection window, we first double the size of the detection to obtain local context. We build image-specific color models of the putative object and its local background using a category-specific shape prior (***middle***). We show in Fig.3 how to automatically learn such priors. We use the prior to compute a weighed color histogram for the object and its local background. These color models are fed into a graphcut algorithm to produce a fg/bg segmentation. We then use the segmentation mask as a feature vector (fed into a linear SVM) to classify the putative detection as a true positive or false positive. We visualize the linear classifier on the* **right***, with light areas corresponding to positive weights and dark areas corresponding to negative weights. The classifier is learned from training data, as shown in Fig.4.*

tation is in general a hard problem, it is much easier with top-down knowledge. The initial detection window defines a rough scale and location of a known object category. This information makes segmentation considerably easier. The resulting segmentation identifies those pixels in the detection window that actually lie on the object.

This in turn allows us to use explicit *shape* cues to verify that an object is actually present. We learn a category-specific *shape classifier* that distinguishes good segmentations from bad segmentations (obtained from false positive windows). We show that such a strategy leads to state-of-the-art performance (10-20% improvement over existing systems) across a number of difficult datasets, such as the Pascal VOC2006 challenge [4], the INRIAPerson database [3], and LabelMe [14].

2. Related Work

There are a number of approaches that combine recognition with segmentation [12, 15, 11, 18, 10]. We focus on segmentation, not as a goal in of itself, but as a mechanism that quantitatively improves recognition performance. In this way, our approach follows the spirit of [15, 10]. However we compute a local figure-ground segmentation rather than a global image parse. Our work is more related in approach to [19] and the concurrent work of [13]. We differ in that we employ shape classifiers to remove false-positives and in our limited use of shape priors during segmentation (as described in Sec.4). Our evaluation component is more involved, since we compare performance across multiple object classes with established baselines on benchmark datasets.

The strategy of detection followed by verification dates back to the *hypothesis-and-test* paradigm for model-based vision [9, 7]. Such algorithms were introduced originally for model alignment. Hypothesized 3D poses are verified with edge features computed from an image. Similar ideas have also proved useful for category-level recognition [12]. These strategies require efficient mechanisms of searching through hypotheses, and accurate tests that verify correct matches. We use scanning-window classifiers as efficient *attention* mechanisms to generate candidate hypotheses, and verify matches with *segmentation* cues.

An important component of our verification stage is a shape model that can distinguish between good and bad segmentations (where bad ones are associated with false positive windows). Shape models in computer vision have a storied history, dating back at least to the deformable models of [5, 6]. Our approach is different than most in that shape is encoded *discriminatively*. Rather than scoring a segmentation using an explicit model of face shape, we learn a classifier for face/non-face segmentations.

3. Overview

To overcome the typical false positives associated with window-based detectors (Fig. 1), we would like to augment the detector with a segmentation cue. This is computationally difficult. As it is, window-based approaches need to be fairly efficient, since one performs an explicit search over hypotheses such as scales and translations. Enlarging the search space to the set of all segmentations seems computationally infeasible. We propose a simple hypothesize-and-test framework to address these computational issues (shown graphically in Fig. 2):

- Given an image, we run a baseline detector tuned to generate many candidates (i.e., tuned for low missed-detections and high false-positives). We typically encounter a few hundred candidates per image.

- At each hypothesized detection, we wish to obtain a local figure-ground segmentation.

  - To exploit top-down knowledge, we use a *category-specific* shape prior to learn *image-specific* color models for the object and the local background. This prior is learned off-line from training data, while the color-models are learned "on-the-fly" at run-time.

  - These color models are used in a graphcut framework to segment the candidate image window into figure/ground.

- We use a *category-specific* shape classifier to label a putative segmentation as a true positive versus a false positive. This classifier is learned off-line from training data.

4. Algorithm

At each hypothesized detection window, we wish to identify those pixels that belong to the object. We first enlarge the detection window to examine the local image context. We do not bother enlarging the window if the baseline detector already uses an enlarged window with context (as does DalalTriggs). We segment the pixels within the enlarged window into object (foreground) versus background using a binary graphcut [1]. A graphcut segmentation minimizes the following energy

$$E(l_1, \ldots, l_K) = \sum_x c(l_x) + \alpha \sum_{x,y \in N} \mathcal{I}(l_x \neq l_y) \qquad (1)$$

where l_x is a binary label for pixel x, $\mathcal{I}$ is the identity function, and N is the set of 4-connected neighbors. The first term is a unary potential that defines to what extent an individual pixel favors a foreground (fg) versus background (bg) label. The second pairwise term defines to what extent neighboring pixels should agree. Here, we use an Ising model [1].

Unary term: To score the unary potential, we need a pixel-based model of the fg and bg. We use a color histogram. We first normalize each detection to a canonical coordinate frame. Assume that for each pixel in given detection window, we have a prior distribution that it will be foreground – we call this $p_{fg}(x)$. We describe in Section 4.1 how to learn such distributions. We use the distribution to calculate a foreground and background histogram color model for a given detection window by

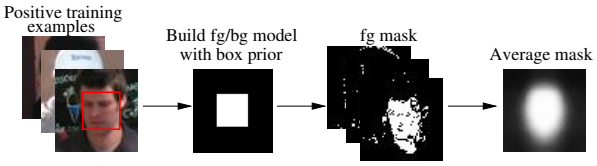

Figure 3. *Learning a shape prior from unsegmented training data. We run our detector on our training set (which known ground-truth face locations). We use the positive detections to learn a shape prior. We build a foreground color histogram from the detected window, and build a background color histogram from a local context window surrounding the detection. We use the color models to classify the pixels as foreground/background. We average the masks across the positive training set to define a shape prior.*

$$\Pr(fg(k)) \propto \sum_x p_{fg}(x)\mathcal{I}(im(x) = k) \qquad (2)$$

$$\Pr(bg(k)) \propto \sum_x (1 - p_{fg}(x))\mathcal{I}(im(x) = k) \qquad (3)$$

where $im(x)$ is the bin number for pixel at location x. We use 16 bins along each of the R,G, and B axes. In theory, the bin index k could vary from 1 to $16^3 = 4096$. In practice, we only consider bins with non-zero entries to avoid numerical issues. Typically, we find only a few hundred distinct colors at each detection window. The unary cost will be negative log probabilities under the model:

$$c(l_x = 1) = -\log(\Pr(fg(im(x)))) \qquad (4)$$

$$c(l_x = 0) = -\log(\Pr(bg(im(x)))) \qquad (5)$$

Role of prior p_{fg}: Typically, shape priors for graphcuts appear explicitly in the unary term (usually in the form of an unsigned distance function to a contour [13, 11]). In our case, they only appear implicitly through the learned fg/bg color models. We opted for this approach because we want a strong low-level signal present in the final segmentation. We do not want the segmentation to *always* look face-like because we could be evaluating a false-positive detection. We want to *discover* whether the resulting segmentation is produced by a face. We do this by using our shape classifier to determine if the segmentation indeed looks face-like. This tension between the role of the prior and utility for verification is likely worth examining further.

Pairwise term: In our model, the pairwise term is a single constant. We used $\alpha = .4$ for all our experiments. Possible extensions would be to make the term by contrast dependent [1] or category-specific (tuned by say, cross validation). We found this was not necessary.

Given the pairwise and unary term, we compute the minimum-energy labeling of pixels as fg/bg, under Eq. 1. This can be efficiently solved with a min-cut max-flow algorithm [1]. This defines a binary segmentation mask that

we rasterize into a feature vector. We then perform linear classification on the feature vector to verify that the hypothesized detection was in-fact correct (Sec. 4.2).

4.1. Learning the shape prior

Our top-down prior for a fg/bg segmentation is a pixel-based independent Bernoulli model. Each pixel x within the detection window has a prior probability $p_{fg}(x)$ of being foreground, and a prior probability of $1 - p_{fg}(x)$ of being background. Given T training images with ground-truth segmentations l^t, the maximum likelihood estimate (MLE) of p_{fg} is the sample average:

$$p_{fg}(x) \stackrel{MLE}{=} \frac{1}{T} \sum_t l_x^t \tag{6}$$

Unfortunately, there does not exist much segmented training data. However, it is common to have *bounding box* locations for positive training examples (as in PASCAL). This allows us to construct a crude *box-prior*:

$$p_{fg}^0(x) = \mathcal{I}(x \in \text{detection window}) \tag{7}$$

We use this initial shape prior to segment the training examples by optimizing Eq.1 with graphcuts. The yields a set of segmentations l^t that we interpret as "ground-truth". Following Eq.6, we compute p_{fg} by averaging them (see Fig.3). This simple approach is similar to the unsupervised algorithm from [17], though it avoids an explicit registration step by leveraging ground-truth bounding boxes.

4.2. Learning the shape classifier

To learn the segmentation-mask classifier, we first apply our window-based detector to images from our training set, where we know ground truth locations of objects. This produces a set of true-positive and false-positive candidate detections.

We then perform the graph-cut segmentation from Eq. 1 on the detected windows, using the prior learned from Eq. 6. This produces a collection of true-positive and false-positive segmentation masks. We define a feature vector for each window consisting of the rasterized segmentation mask appended by the raw score of the baseline detector:

$$f_{window} = \begin{bmatrix} l_1 \dots l_K & r_{det} \end{bmatrix} \tag{8}$$

We learn a *linear* SVM that separates the true and false-positives. We tried various other nonlinear extensions (*e.g.*, a Nearest Neighbor classifier, a Gaussian kernel SVM), but saw no significant improvement. The linear model also provides an intuitive understanding of what the shape cues the classifier is using. For example, from Fig.4, it is clear that the classifier learns to penalize foreground pixels above the head and beside the lower jaw. The classifier also favors foreground pixels at the forehead, neck, and ears. These simple constraints help separate true faces from false positives (from textured regions such as trees).

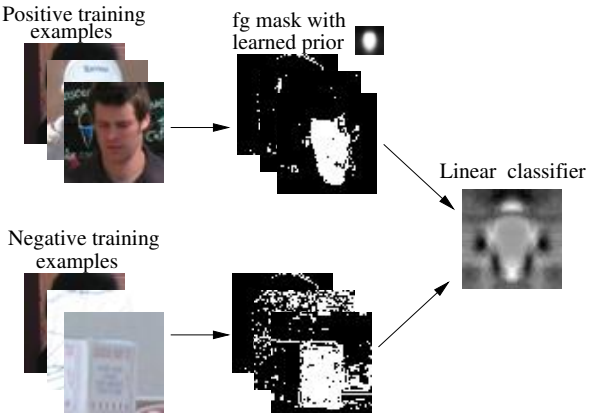

Figure 4. *Learning a shape classifier. We first run our baseline detector on training images with known ground truth locations. This yields a set of positive and negative detections. We learn* image-specific *fg/bg color models for each detection, using the the* shape prior *defined in Fig. 3. We use these color models to segment each detection window, mimicking the run-time algorithm from Fig.2. We then interpret the segmentation masks as features, and learn a linear classifier the separates the positive training examples from the negatives. We visualize the learned* shape classifier *on the* **right**, *with light areas corresponding to positive weights and dark areas corresponding to negative weights. In this case, the classifier favors segmentations that seem to capture the forehead and neck. The classifier also penalizes foreground pixels above the head and to the side.*

We show results in Fig.5 for applying the same learning procedure to a window-based classifier for pedestrians and cars. To learn a pedestrian shape-prior, we apply the procedure from Fig. 3 to images of pedestrians. To learn a pedestrian shape-classifier, we apply the procedure from Fig. 4 to both positive and negative training images of pedestrians. We similarly repeat for the car detector. The pedestrian shape classifier favors foreground pixels at the head and penalizes foreground regions above the head. This constraint helps removes false positives from long cylindrical objects such as lamp posts. The car shape classifier penalizes foreground pixels to the left and right of the car. This removes false positives due to objects with long horizontal edges.

5. Results

We tune our baseline detection systems to yield low missed detections and high false positives. For ViolaJones, we prematurely truncated the last 5 stages of the cascade of classifiers from the OpenCV implementation [2]. For Dalal-Triggs, we lowered the bias threshold in the linear SVM by 1. The shape priors and shape classifiers for ViolaJones frontal/profile face detector were trained from the LabelMe

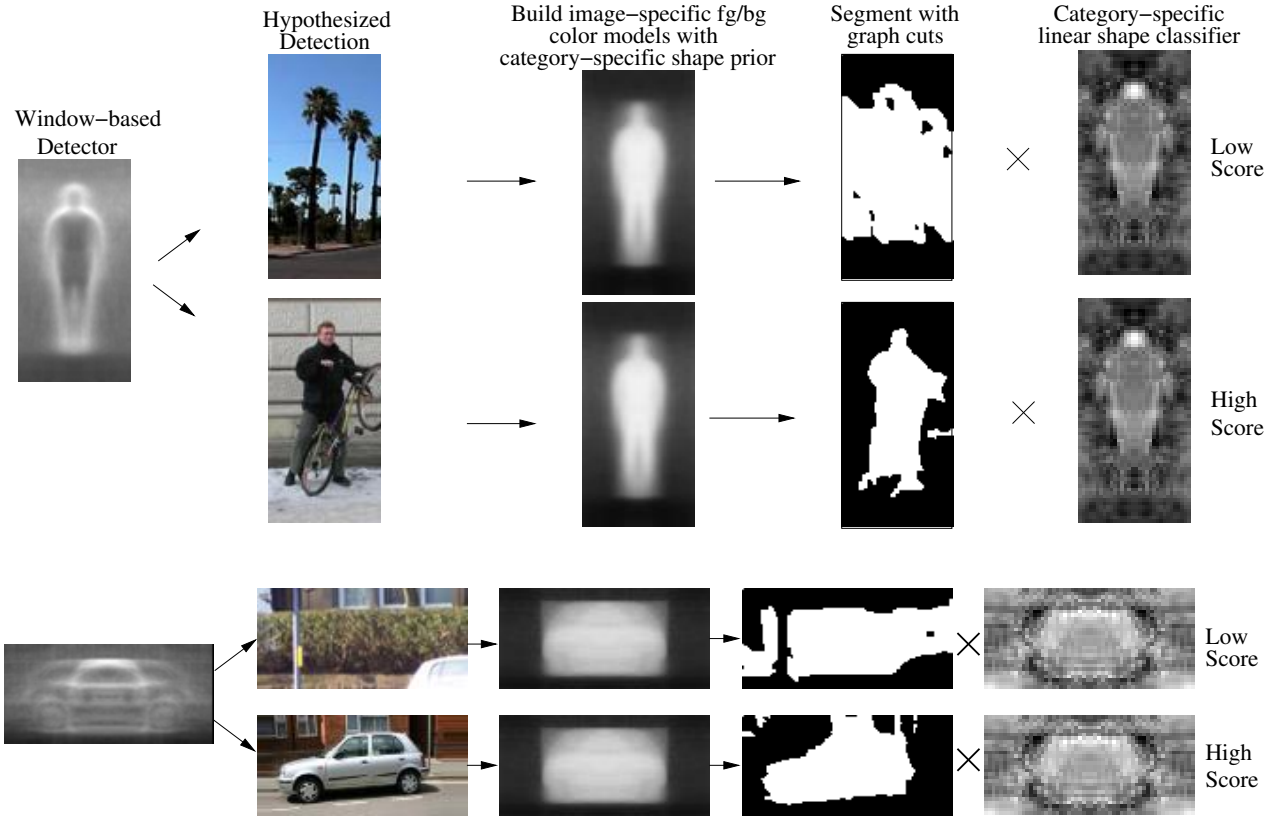

Figure 5. *Algorithm for segmentation-based verification applied to people and car detection. Since the baseline window-based detector includes local image context, we do not enlarge the window after detection (as in Fig. 2). The people classifier seems to learn the foreground pixels along the head are important, and heavily penalizes foreground pixels above the head and bordering the legs. This removes false positives with long vertical edges, such as pillars. The car classifier penalizes foreground pixels to the left and right of the putative car detection. This eliminates false positives due to long horizontal edges, such as shrubs.*

training set. The shape prior/classifier for the DalalTriggs person detector was trained on the INRIAPerson training set. Finally, the prior/classifier for the DalalTriggs car detector was trained using the PascalCar training set. When learning the prior/classifier, we use the same canonical window size as the baseline detector (except for ViolaJones, we used twice the size due to to the enlarged window context).

Scoring: For all our experiments, we precisely followed the evaluation methodology outlined in the PASCAL challenge [4]. A putative detection is considered true if the area of overlap between the predicted window and the ground truth window exceeds 50% of the area of the union of the two windows. At most one putative detection can be assigned to a given ground truth object – any additional overlapping detections are labeled as false. As in the PASCAL challenge, we summarize performance with the average precision (AP) computed from a precision recall (PR) curve (see Table 1). Following conventions, it is computed by averaging the precision at recall values spanning $\{0, .1, .2, \ldots, 1\}$.

LabelMeFaces: Since our procedure relies on computing a segmentation using color cues, we cannot use standard benchmarks for face detection (which are grayscale). Instead, we collected all the face and head detections from LabelMe (as of Aug 2006), re-verifying all detections manually. This yields a quite large and varied dataset of 1,186 images containing 2,184 faces (see Fig. 9). We split the images randomly into a train and test set. We use the training set to learn a separate shape prior and shape classifier for frontal and profile faces. We use the test set to evaluate both frontal and profile face detection.

We plot the PR curve for the LabelMe test set in Fig. 6. Our segmentation-based verification stage improves performance by 8.9% for frontal face detection, and 15.7% for profile detection. These results are impressive given the quality of our baseline ViolaJones detector. Segmentation particularly improves profile detection. Profile face detection is known to be hard because the face contour plays a crucial role. Typical frontal face detection algorithms focus on internal edges, which are more reliable to extract. Our segmentation procedure explicitly recovers the face/nonface

boundary (see Fig.9), and furthermore the segmentation-mask classifier explicitly exploits it as a feature (see Fig. 2).

Pascal2006Cars: We show results using the car images from Pascal2006. We use the designated training images to learn the segmentation prior and classifier. We plot the PR curve for the test set in Figure 7. The second row in Figure 9 shows some image results. The average precision for our algorithm (.538) clearly outperforms the state-of-the-art of .444 [4].

Pascal2006Person: We show results from the people images in PASCAL. This is a very challenging dataset, with current leading methods yielding a low AP of .164 [4]. Our procedure again improves the state-of-the-art, yielding a score of .188. We train our system using the training images from the INRIAPerson dataset. We show example test images in the third pair of rows from Figure 9. We see that many of the missed detections arise from people surrounded by clutter around their lower body. Scenes involving people riding motorcycles or horses prove difficult for a template-based detector, since the edge responses generated by those objects could confuse the detector. Again, explicit reasoning about segmentation helps in these challenging cases.

INRIAPerson: We plot the PR curve for testset performance in Fig.7. We obtain an AP of .774, which is considerably better than the default score of .708 we obtained running the raw detector. The last pair of rows from Figure 9 shows some image results. Here, missed detections often arise from non-standard body poses. This suggests that a more sophisticated shape prior/classifier would help.

LabelMePerson and LabelMeCars: We also performed experiments using the LabelMe test data from [8]. We use the car and pedestrian detector trained from the PascalCar and INRIAPerson training images. Here, using all the ground truth labels in the test set, our segmentation procedure provides very little improvement - 2.2% for people and 6.0% for cars. When scoring large people and cars (of a scale equal to the minimum scale of the baseline detector), we see a noticable improvement of 8.42% and 12.5% respectively. This data is harder than our other datasets since people and cars tend to occur at smaller scales in the street scenes. This suggests that it is indeed difficult to recognize small objects using local information – as eloquently argued in [8]. However, at larger scales, we argue (and demonstrate) that local segmentation cues are still quite useful.

6. Discussion

Looking at Table 1, we tend to see a performance boost between 10-20% across our datasets. This is impressive given the quality of our baseline detectors. One might argue the success of our approach comes from the use of color, which is often ignored as a cue for recognition. For example, our baseline detectors work mostly with grayscale data. This is because color as a high-level object cue is not particularly useful - not all cars are red. However, color can be extremely helpful as a *low-level* segmentation cue – if a particular car is red, we can identify car image regions by looking for nearby red pixels. This in turn allows us to use high-level *shape* cues for recognition.

We posit our argument applies to other low-level cues such as grayscale intensity and texture. One important caveat is that such an approach works best for medium to large-scale detections - in our case, we observe that the minimum dimension of a detection needs to be roughly 30-40 pixels. When objects are smaller, it is likely that a large spatial context plays a crucial role in recognition.

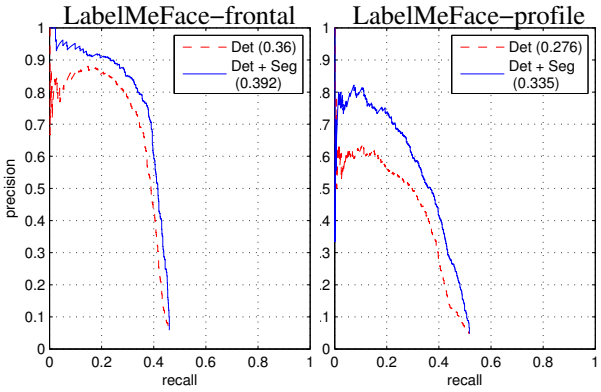

Figure 6. *PR curves for face detection on 1186 images from LabelMe. On the* **left**, *we score performance for the ViolaJones frontal-face detector implemented in OpenCV. On the* **right**, *we score performance for the profile detector implementation. Our segmentation-based verification improves frontal face detection by 8.9%, and improves profile detection by 15.7%. Our segmentation stage helps more in the high-precision/low-recall regime, and proves to be quite effective for profile faces. This is because our face segmentations tend to capture the face contour (see Fig.9), which likely plays a useful role in profile face detection.*

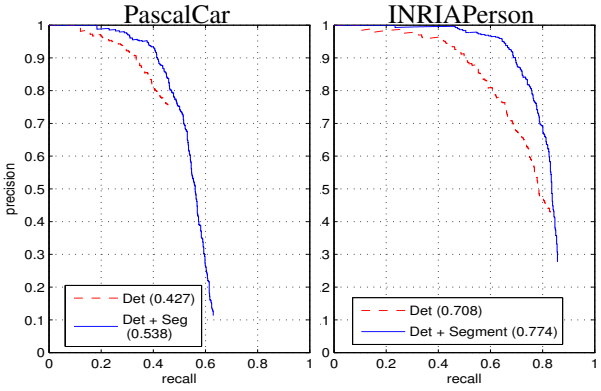

Figure 7. *PR curves for the DalalTriggs detector augmented with a segmentation-verification step. On the* **left**, *we show results on the PascalCar dataset. Our AP of .538 is considerably higher than the best reported result of .444 [4]. On the* **right**, *we show results on the INRIAPerson dataset. Our AP of .774 is also noticeably better than the score of the initial detector (.708).*

Dataset	Det	Det+Seg	% increase
LabelMeFace-Frontal	.360	.392	8.9
LabelMeFace-Profile	.276	.335	15.7
Pascal-Car	.427	.538	26.0
Pascal-CarBig	.496	.644	29.9
LabelMe-Car	.369	.391	6.0
LabelMe-CarBig	.457	.514	12.5
INRIA-Person	.708	.774	9.3
Pascal-Person	.171	.188	9.9
Pascal-PersonBig	.253	.294	16.2
LabelMe-Person	.316	.323	2.2
LabelMe-PersonBig	.582	.631	8.42

Table 1. *Average Precision (AP) results across many datasets.* **Det** *refers to the baseline detector, while* **Det+Seg** *refers to our approach of detection followed by segmentation-based verification. The 'Big' datasets count a ground truth object location if it is not truncated and larger than the minimize size of the baseline detector. We score Person and Car detection on LabelMe using the dataset from [8]. Our results on PascalCars (.538) and PascalPerson (.188) prove favorable to the best-reported results of .444 and .164 [4]. Generally speaking, our segmentation-based verification tends to boost performance from 10-20%, with a larger increase for the 'Big' datasets.*

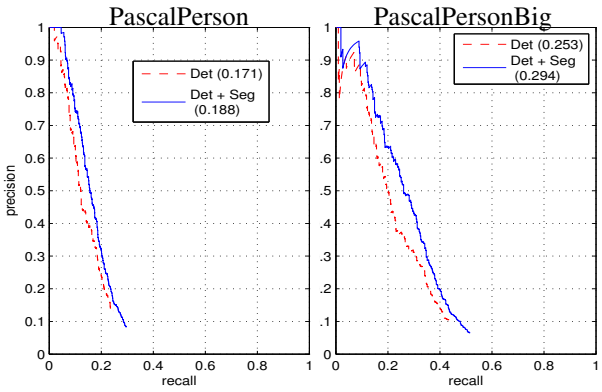

Figure 8. *PR curves for person detection on PASCAL 2006. On the* **left***, we show results on the original testset. On the* **right***, we show results on an 'Big' version of the dataset that only scores large un-truncated objects. On the Big dataset, we show a large improvement of 16.2%.*

Broadly speaking, we argue that existing object detection systems operating in the mid to large-scale regime can be improved by incorporating low-level segmentation cues. We have proposed a simple and efficient hypothesize and test mechanism for implementing such an agenda. We have rigorously demonstrated state-of-the-art performance, improving performance 10-20% over established baselines on on several benchmark datasets.

Acknowledgments: Thanks to Navneet Dalal and Derek Hoiem for discussions and code and Xiaofeng Ren for his ViolaJones implementation.

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV (1)*, pages 377–384, 1999.

[2] G. Bradski, A. Kaehler, and V. Pisarevsky. Learning-based computer vision with intel's open source computer vision library. In *Intel Technology Journal*, 2005.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[4] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual object classes challenge 2006 (voc2006) results. In *Selected Proceedings of the Second PASCAL Challenges Workshop*, 2006.

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

[6] U. Grenander, Y. Chow, and D. Keenan. *Hands: a pattern theoretic study of biological shapes*. Springer-Verlag, 1991.

[7] W. Grimson and D. Huttenlocher. On the verification of hypothesized matches in model-based recognition. In *European Conference on Computer Vision*, pages 489–498, 1990.

[8] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, June 2006.

[9] D. Huttenlocher and S. Ullman. Object recognition using alignment. In *ICCV*, page 102, 1987.

[10] C. R. J. Shotton, J. Winn and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for mulit-class object recognition and segmentation. In *ECCV*, 2006.

[11] M. Kumar, P. Torr, and A. Zisserman. Objcut. In *CVPR*, 2005.

[12] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC*, 2003.

[13] J. Rihan, P. Kohli, and P. H. S. Torr. Objcut for face detection. In *ICVGIP*, pages 576–584, 2006.

[14] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. Technical report, MIT, 2005.

[15] Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *Proc ICCV*, 2003.

[16] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[17] J. Winn and N. Joijic. Locus: Learning object classes with unsupervised segmentation. In *ICCV*, 2005.

[18] S. Yu, R. Gross, and J. Shi. Concurrent object recognition and segmentation with graph partitioning, 2002.

[19] L. Zhao and L. Davis. Closely coupled object detection and segmentation. In *ICCV*, pages 454–461, 2005.

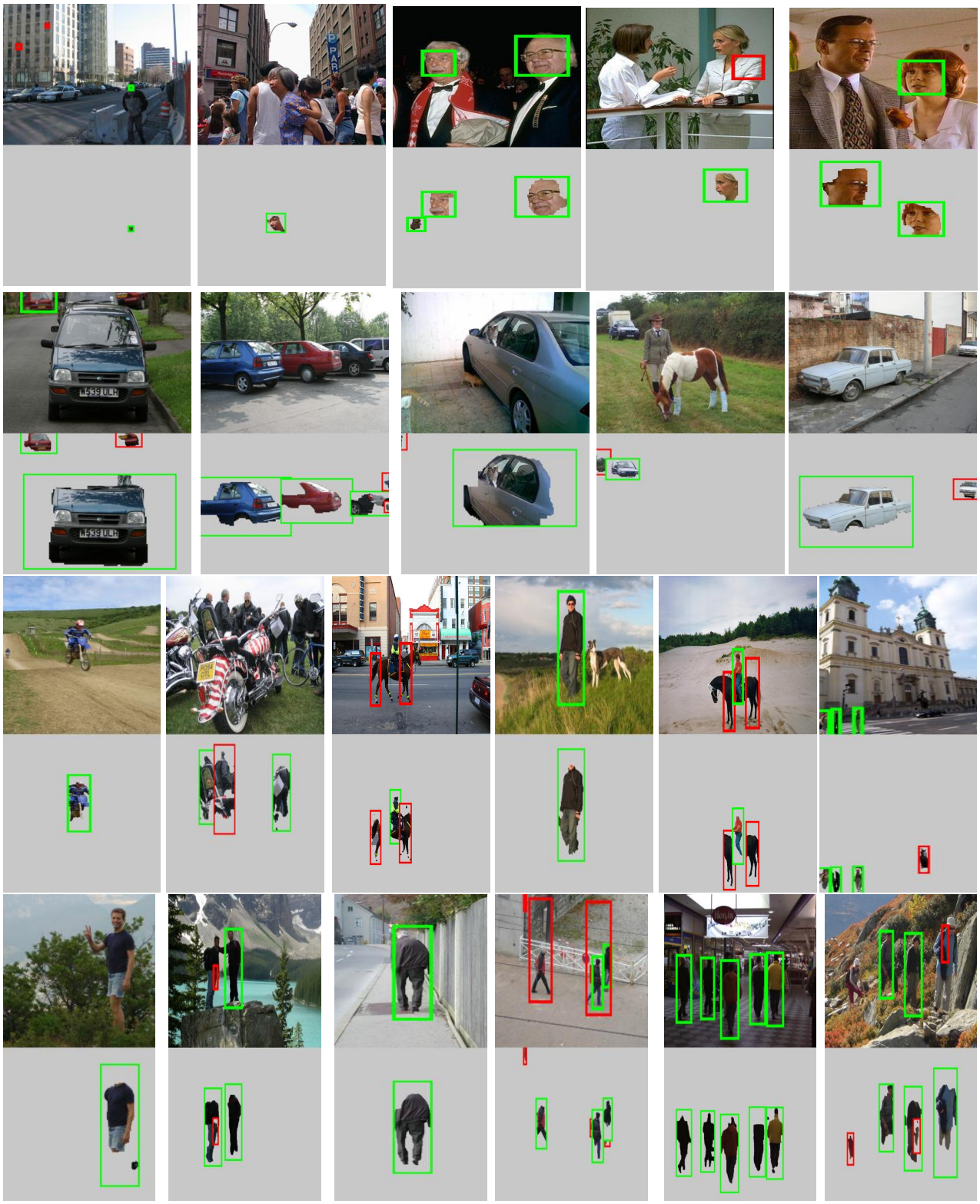

Figure 9. *Results on test images from various datasets. For each pair of rows, the top row displays results from the original window-based detector. The bottom row shows detections obtained with a segmentation-verification step. Since we compute an explicit segmentation, we visualize that as well. The green boxes denote true positive detections, while the red boxes denote false positives. The* **top** *pair of rows show results from the LabelMe face dataset. The* **second** *and* **third** *rows show results from Pascal VOC2006 dataset (for cars and people, respectively). The* **forth** *row shows results from the INRIAPerson c*