A Hybrid Object-Level/Pixel-Level Framework For Shape-based Recognition

Owen Carmichael and Martial Hebert

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

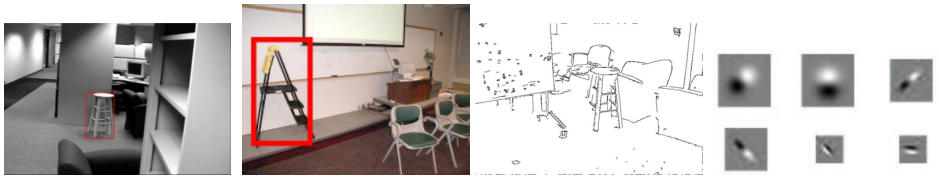
This paper presents a technique for shape-based recognition that fuses pixel-level and object-level approaches into a unified framework. A pixel-level algorithm classifies individual pixels as belonging to a target object or clutter based on automatically-selected shape features computed in a spatial arrangement around them; an object-level algorithm classifies object-sized rectangular image regions as objects or clutter by aggregating pixel classifier scores in the regions. We train a cascade of interleaved pixel-level and object-level modules to quickly localize complex-shaped objects in highly cluttered scenes under arbitrary out-of-image-plane rotation. Experimental results on a large set of real, highly-cluttered images of a common object under arbitrary out of image plane rotation demonstrate improvements over cascades of strictly pixel-level modules.

1 Introduction

Object recognition algorithms have made great strides in recent years, leading to techniques capable of robust, real-time recognition of certain types of objects such as faces, cars, and buildings [23][19]. However, recognizing objects with complex shape characteristics such as holes and networks of thin linear structures (*e.g.* the legs and supports on the stool and ladder in Figure 1(a)) remains challenging. In this paper, we present an efficient technique for using example images of a particular complex-shaped object in typical environments to automatically select shape features and train a classifier cascade to localize that object in highly cluttered novel views under arbitrary out-of-image-plane rotation [1]. Figure 1(a) shows two typical results. Recently, several *pixel-level* algorithms have successfully addressed the problem of using local shape features to estimate whether image pixels correspond to an instance of a target object, or to clutter [4][14][1]. Like local patch-based recognition techniques (*e.g.*, [17]), they need to apply a separate *object-level*

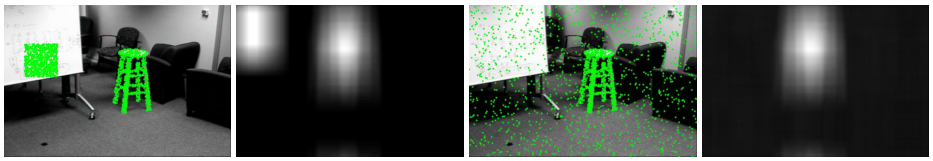
[1] We note that it is possible to extend our technique to handle object scale variations either by processing the same image repeatedly at a variety of scales [19] or by rectifying image features to a canonical scale [12][8]. We also note that we focus on detecting a single, individual instance of a wiry object across a broad variety of viewing conditions because this problem is extremely challenging and largely unsolved; we are confident that solutions to the more general problem of detecting entire classes of wiry objects will build on advances made toward detecting individual wiry objects

BMVC 2004 doi:10.5244/C.18.99



(a) (b) (c)

Figure 1: Figure 1(a): Example results from the technique presented in this paper are shown in red. Figure 1(b): A poorly-tuned edge detector misses edges on the rear legs of the stool. Figure 1(c): 6 edge operators selected during feature selection.



(a) (b)

Figure 2: Motivating example for optimizing pixel-level and object-level processes in a unified procedure. Figure 2(a) left and Figure 2(b) left: An image is processed by two hypothetical pixel-level classifiers whose performance at a pixel level is identical– both mistakenly label the same number of pixels on the stool "clutter" and the same number of pixels on the clutter "stool" (stool pixels are in green). Figure 2(a) right and Figure 2(b) right: stool pixels are clustered into groups according to their location in the image; each pixel represents the number of stool pixels in a stool-sized image rectangle with an upper lefthand corner at that pixel, with white representing a high number of stool pixels. In the latter case, it is easy to localize the stool, while in the former case a false positive is triggered. A pixel-level classifier trained with no object-level feedback has no way to prefer the former situation to the latter.

algorithm to assemble individual pixel detections into overall descriptions of what objects are present in the image, and what their properties (*e.g.* pose and spatial extent) are. If detailed, canonical 2D or 3D models of overall object shape or configuration are available, it is possible to align object pixels to them [1][9][8]; otherwise, it is necessary to cluster individual local detections [13] to localize the object to an image region, or use voting [17] to determine more generally whether the object is present. This paper provides a unified framework for shape-based recognition that avoids the need to optimize separate pixel-level and object-level processes. Figure 2 illustrates the importance of training pixel-level recognition techniques in an object-level context. As part of this process, we automatically select salient shape features to extract directly from the raw image, instead of relying on a separately-tuned binary edge detector.

Figure 5, top right, gives a block diagram of how our algorithm processes a novel image by executing a cascade of interleaved pixel-level and object-level computational phases. The first pixel-level phase classifies pixels based on the spatial arrangement of edge features in small local neighborhoods surrounding them. The first object-level phase aggregates the classification scores for all pixels in a rectangular image region into an overall score for that region, and discards pixels in low-scoring regions. Then, pixels in high-scoring regions are passed to a second pixel-level phase, which re-classifies the remaining pixels based on edge features computed over larger local neighborhoods. A second object-level phase aggregates those pixel scores, discards low-scoring image regions, and so on. The pixel-level phases quickly classify pixels as object or clutter based on local shape properties; the object-level phases remove from consideration large image regions whose pixels, taken together, do not indicate the presence of the object. Since image regions are discarded as soon as they are given a low score, our cascade approach focuses computation on the parts of the image which appear most like the target object and require more extensive local feature computation to classify.

The rest of this section surveys related work. We present our local shape features, pixel-level classifiers, and object-level classifiers in Sections 2, 3 and 4 respectively. In

Section 5 we fuse the two types of classifiers into a single optimization framework, and Section 6 discusses an efficient cascade structure for applying the classifiers to images. Experimental results on hundreds of highly-variable images of a stool are in Section 7, and we conclude in Section 8.

Related Work

We refer to our recognition approach as a *hybrid* method because it optimizes both object-level and pixel-level processes. A related approach [20] alternates between data-driven and model-driven steps to segment and categorize image regions into face, text, and other categories in a generative MCMC framework. Another related approach is due to Yu and Shi [25], who provide an algorithm for fusing object-level and pixel-level affinity measures into a single optimization for segmentation.

The common theme of related work in local shape-based recognition [4][14][1] is the use of local distributions of edges or edge-based shape features as a cue for estimating whether an edge pixel corresponds to a target object or clutter. These methods successfully estimate a set of pixels which project onto a complex-shaped object of interest, and assemble the individual pixels into object-level detections by applying a separate alignment [1] or grouping [4] algorithm. Meanwhile, recent work in cascade architectures for object-level recognition (*e.g.,* [23][10][24]) has exploited the key insight that the amount of computation required to classify most image regions as "object" or "clutter" can be extremely low, and that dramatic speedups can be gained at run time by saving computation for the "rare event" of image regions that look very similar to object instances. Our approach dramatically improves on our previous pixel-level shape recognition technique [4] by automatically fusing local shape-based methods [4][14][1] with object-level cascades [23][10][24] to quickly discard background image regions.

Furthermore, we improve on the previous local shape-based techniques by automatically selecting shape features to extract from the raw image instead of requiring that binary edges be extracted prior to recognition. Algorithms phrased in terms of analyzing binary edge pixels have no chance to detect portions of the object if a poorly-tuned edge detector fails to detect edges on corresponding portions of the image; consider, for example, the missing rear legs of the stool in Figure 1(b). Our feature selection technique builds an initial set of non-redundant features, as in [16] for example, and selects discriminating features from this set in a way similar to several recent algorithms [22][11][24][6].

2 Image Features

Since we assume that our target objects are well-characterized locally by their shape, we classify each pixel $\mathbf{q} = [x, y]$ based on the values of edge operators evaluated in the neighborhood of $\mathbf{q}$. We draw the edge operators from the family of first derivative-of-Gaussian (D-of-G) operators since they are straightforward to parameterize in terms of their scale and orientation characteristics. An *edge operator probe* $eop(\mathbf{p}, I, \theta)$ at *probe center* $\mathbf{p}$ in image I convolves the image at $\mathbf{p}$ with a D-of-G operator g_θ, that is,

$$eop(\mathbf{p}, I, \theta) = \sum_{\mathbf{t}} g_\theta(\mathbf{t}) \cdot I[\mathbf{p} - \mathbf{t}] \quad , \quad g_\theta(\mathbf{t}) = \frac{-x(\mathbf{t}, \phi)}{\sigma_x^2} * \exp(-\frac{1}{2}(\frac{x(\mathbf{t}, \phi)^2}{\sigma_x^2} + \frac{y(\mathbf{t}, \phi)^2}{\sigma_y^2}))$$

where $\theta = [\phi, \sigma_x, \sigma_y]$ represents the scale (σ_x,σ_y) and orientation (ϕ) response characteristics of g_θ, and $x(\mathbf{t},\phi) = \mathbf{t} \cdot \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix}$ and $y(\mathbf{t},\phi) = \mathbf{t} \cdot \begin{bmatrix} -\sin(\phi) \\ \cos(\phi) \end{bmatrix}$. An edge operator probe responds strongly to the presence of edges at orientation ϕ and spatial scales described by σ_x and σ_y in the vicinity of $\mathbf{p}$.

For local shape features, we select a discrete set of D-of-G operators $\{g_{\theta_1} \cdots g_{\theta_n}\}$, and classify image pixels $\mathbf{q}$ by evaluating edge operator probes $[eop(\mathbf{q}+\delta_1, I, \theta_1), \cdots, eop(\mathbf{q}+\delta_m, I, \theta_1), \cdots, eop(\mathbf{q}+\delta_1, I, \theta_n), \cdots, eop(\mathbf{q}+\delta_m, I, \theta_n)]$ at a set of *shifted probe centers* $\{\mathbf{q}+\delta_1, \cdots, \mathbf{q}+\delta_m\}$ laid out over a local neighborhood, or *aperture*, surrounding $\mathbf{q}$. We call the offsets $\{\delta_1, \cdots, \delta_m\}$ *relative probe centers*. These image features represent how the edge characteristics captured by the operators are spatially arranged over an aperture surrounding $\mathbf{q}$ by sampling the operator responses at $\{\mathbf{q}+\delta_1, \cdots, \mathbf{q}+\delta_m\}$. In Sections 2.1 and 3.2 we describe how to automatically select a set of edge operators that are relevent for recognition; in Section 6 we describe how to determine the aperture size. By automatically selecting shape features, we determine which edge orientations and scales are relevant for recognition rather than implicitly selecting relevant edge characteristics by hand through the tuning the parameters of a binary edge detector.

2.1 Feature Selection 1

There are infinitely many possible edge operators g_θ. Therefore, we need to select a small, discrete set of edge operators to use in our image features. This feature set should discriminate object pixels from clutter pixels when evaluated at shifted probe centers, and for efficiency the features in the set should not produce redundant responses on images of interest. This section addresses the redundancy issue by building a set of edge operators which produce distinct responses when applied to training images; Section 3.2 describes a technique for subsequently selecting discriminating operators from that set.

Let Φ represent an initial set of orientations sampled uniformly from $[0, \pi]$. Also, let Σ_x represent an initial set of edge operator scales in the x direction, sampled uniformly from an interval of reasonable scales $[\sigma_{min}, \sigma_{max}]$. Σ_y is an analogous set of scales in the y direction. Our approach is to reduce a large *candidate set* of edge operators $\mathscr{G}_1$, containing one edge operator g_θ for each $\theta \in \Phi \times \Sigma_x \times \Sigma_y$, to a *non-redundant* set $\mathscr{G}_2$ of edge operators each of whose responses to a set of training images are distinct from the responses of all other operators in $\mathscr{G}_2$.

For simplicity, we use a greedy forward selection algorithm [2] to reduce $\mathscr{G}_1$ to $\mathscr{G}_2$. A pair of edge operators $g_{\theta 1}$ and $g_{\theta 2}$ is redundant if they produce highly similar operator responses when applied to each of the training images $\{T\}$; we measure redundancy by normalized correlation of the responses. Starting with an empty $\mathscr{G}_2$, we add candidate edge operators $g_{\theta 1} \in \mathscr{G}_1$ to $\mathscr{G}_2$ if and only if $corr(g_{\theta 1} \circ T, g_{\theta 2} \circ T)$ is less than a threshold t for all T and all $g_{\theta 2} \in \mathscr{G}_2$. Figure 1(c) shows a sample of 6 of the 32 edge operators selected for the non-redundant set from a candidate set of 640 operators, based on responses to a set of 50 training images of the stool and $t = .8$.

3 Pixel-Level Classifiers

This section briefly summarizes our pixel-level classifiers, which follow the general approach of [4] in using decision trees to compute a sparse set of local features over an

aperture.

3.1 Decision Trees

Figure 3: Illustration of the image processing steps during one cascade phase for the pixel-level cascade (top row) and pixel-and-object-level cascade (bottom row). Top row, left to right: the input image, pixel scores (whiter pixels have higher scores), and thresholded pixel scores (white pixels pass the threshold). Bottom row, left to right: pixel scores, aggregation scores (each pixel corresponds to the upper left corner of a rectangular region), thresholded aggregation scores, and pixels within the regions passing the threshold.

Given a pixel $\mathbf{q}$ in a novel image, we compute the responses of a set of edge operators $\mathscr{G}$ to image locations at $\{\mathbf{q}+\delta_1,\cdots,\mathbf{q}+\delta_m\}$ for a set of relative probe centers $\Delta=\{\delta_1,\cdots,\delta_m\}$. Our goal is to train a classifier which, at run time, classifies $\mathbf{q}$ as an object or clutter pixel based on those image features. More specifically, at training time we want to tune a classifier to maximize the scores it gives to object pixels, and minimize scores it gives to clutter pixels, in a set of labeled example images. An additional goal is run-time efficiency, *i.e.* fast classification by computing as few image features as possible.

We employ decision trees to meet these goals. Each node in the tree evaluates an edge operator probe $eop(\mathbf{q}+\delta_1, I, \theta_1)$ where (g_{θ_1}, δ_1) are the edge operator and relative probe center, or *operator-offset pair*, associated with the node. Starting with the root node of the tree, we evaluate a sequence of edge operator probes as specified by the tree until a leaf node is encountered. Associated with each leaf node is a score w for the pixel belonging to the object. By thresholding w, we would be able to make a binary decision about whether $\mathbf{q}$ should be labeled as an object pixel or clutter pixel as in traditional binary decision trees [18]. However, in Section 4, we describe how we instead sum w over rectangular image regions to classify the region as an entire object instance or clutter.

We train the decision trees using a two-step process of tree generation and pruning [18]. After splitting the training images into a *tree-growing set* and a *holdout set*, a decision tree with high classification accuracy on the tree-growing set is induced; the scores associated with the leaves are estimated by counting the number of object pixels and clutter pixels in the tree-growing set assigned to the leaf. In Section 5 we describe how we prune the tree to optimize an object-level performance criterion.

3.2 Feature Selection 2

Training decision trees to discriminate object from clutter pixels using all operator-offset pairs in $\mathscr{G}_2 \times \Delta$ as potential features is usually infeasible due to the large number of features. Therefore, prior to training each pixel classifier, we select a small set of operator-offset pairs that discriminate object pixels from clutter pixels. To do so efficiently, we

(a) (b)
Figure 4: Example images of the stool: low internal clutter (Figure 4(a)), higher internal clutter (Figure 4(b)).

employ a filter method [2] which makes predictions about the usefulness of an operator-offset pair by quickly training a decision stump to discriminate pixels based on that operator-offset pair alone. That is, for each operator-offset pair (g_θ, δ), we use standard decision tree splitting criteria (see, *e.g.*, [15]) to find a threshold that discriminates $\{eop(\mathbf{q}_+ + \delta, T, \theta)\}$ from $\{eop(\mathbf{q}_- + \delta, T, \theta)\}$, for object pixels $\mathbf{q}_+$ and clutter pixels $\mathbf{q}_-$, and to assign a score to the quality of the split. We then train the full decision tree using the k highest-scoring operator-offset pairs. Since we select the features using the decision tree splitting criterion, we at least know that they discriminate well as root nodes in the tree. Our assumption is that as decision tree induction recursively partitions the tree-growing set, those features will continue to discriminate well.

4 Object-Level Classifiers

We use a fast, simple technique for identifying gross regions as whole objects, or clutter, by adding the pixel classifier scores in those regions. Given a novel image I, the pixel classifier assigns a continuous score w to each $\mathbf{q} \in I$. Consider the image W, where $W[\mathbf{q}] = w$. An *aggregation filter* sums the classifier scores over an image rectangle $b_\theta = [b_{\theta w} b_{\theta h}]$ of width $b_{\theta w}$ and height $b_{\theta h}$ centered at $\mathbf{q}$:

$$ag(b_\theta, \mathbf{q}, W) = \sum_{x \in \mathscr{X}, y \in \mathscr{Y}} W[\mathbf{q} + \begin{bmatrix} x \\ y \end{bmatrix}] \quad , \quad \mathscr{X} = [\frac{-b_{\theta w}}{2}, \frac{b_{\theta w}}{2}] \quad , \quad \mathscr{Y} = [\frac{-b_{\theta h}}{2}, \frac{b_{\theta h}}{2}]$$

In order to classify a $b_{\theta w}$-by-$b_{\theta h}$ image rectangle centered at $\mathbf{q}$ as an object instance or clutter, we apply the pixel classifier to the pixels in the rectangle and threshold the *aggregation score* $ag(b_\theta, \mathbf{q}, W)$. Aggregation filters do not reason about how the pixel scores are distributed spatially(as in, for example, [19]), but they are extremely fast to apply at run-time and their detection behavior is controlled by a single acceptance threshold t on the aggregation score. In our experiments, $b_{\theta w}$ and $b_{\theta h}$ are the average height and width of the bounding boxes around the target object in all training images.

5 Joint Training of Pixel-Level and Object-Level Steps

Here, we describe how to jointly train a pixel classifier and aggregation filter in a unified optimization so that the number of *object false positives*– image rectangles covering the clutter that are mistakenly identified as object instances– is minimized, while the number of *object true positives*– image rectangles covering the object that are correctly identified as object instances– is maximized. Since aggregation scores are sums of pixel scores, we meet our object-level goals by maximizing scores for object pixels and minimizing them for clutter pixels. Together, the jointly trained pixel classifier and aggregation filter are referred to as a *pixel-object pair*.

We embed our training goals into the tree-generation and pruning process of decision tree learning; tree-generation optimizes pixel-level classification scores and pruning optimizes object-level aggregation scores. More specifically, given a tree-growing image set whose pixels have been partitioned into object pixels $\{\mathbf{q}_+\}$ and clutter pixels $\{\mathbf{q}_-\}$, we grow a decision tree to produce high pixel classifier scores for $\{\mathbf{q}_+\}$ and low scores for $\{\mathbf{q}_-\}$. Applying the pixel classifier to each pixel in the holdout images leads to a pixel score image W for each holdout image; applying an aggregation filter to W leads to two sets of aggregation scores, one $\{ag(b_\theta, \mathbf{q}_+, W)\}$ for object rectangles and another $\{ag(b_\theta, \mathbf{q}_-, W)\}$ for clutter rectangles. By applying a threshold t to the aggregation scores, we make a binary decision on whether an image rectangle corresponds to an object instance or clutter. An *ROC grading criterion* assigns an overall numerical grade to the set of aggregation scores; in essence, the ROC grade is high if it is easy to find a setting for t that leads to low numbers of object-level false positives and false negatives (*i.e.*, $ag(b_\theta, \mathbf{q}_-, W) > t$ and $ag(b_\theta, \mathbf{q}_+, W) < t$). See [3] for details.

Pruning a subtree from the decision tree changes some of the pixel scores in W, which in turn modifies some of the aggregation scores, which in turn affects the ROC grade. In this way, pruning based on the ROC grade gives us a means for tuning the pixel-level classification tree for better object-level performance. Therefore, we prune the pixel-level decision tree in a greedy, bottom-up way whenever doing so increases the ROC grade. After growing the tree for pixel-level performance and pruning it for a higher object-level ROC grade, the ROC grading criterion guides the selection of an aggregation score threshold t. Applying the pixel-object pair to a novel image consists of using the decision tree to assign scores to all pixels, computing aggregation scores for all image rectangles, and discarding all rectangles with aggregation scores lower than t.

6 A Cascade Of Pixel-Object Pairs

Our overall system is a cascade of pixel-object pairs. The pixel-object pairs repeatedly classify image pixels, aggregate the pixel scores over image rectangles, and discard the pixels in rectangles that have low aggregation scores. As in [4], each successive cascade phase computes edge operator probes over successively larger apertures. More formally, consider a set of relative probe centers Δ which cover a circular aperture, and let $r(\Delta)$ be the radius of the circle. We train a series of pixel-object pairs such that successive pixel classifiers evaluate edge operator probes at successive sets of relative probe centers $\{\Delta_1, \Delta_2, \cdots, \Delta_k\}$ where $\{r(\Delta_1) < r(\Delta_2), \cdots, < r(\Delta_k)\}$. In this way, we grow the aperture size to gradually incorporate more neighborhood information and discard low-scoring regions as soon as the aperture size is large enough to discriminate their pixels as object or clutter. Doing so saves computation for the relatively rare image regions which look similar to the target object. A block diagram of the cascade steps is in Figure 5 (top right) and an illustration of applying one cascade phase to a test image is in Figure 3, bottom row.

7 Experiments

To validate this technique, we experimented in depth with the "stool" images from the WORD image database [5] (Figure 4). The stool rotates to an arbitrary rotation on the

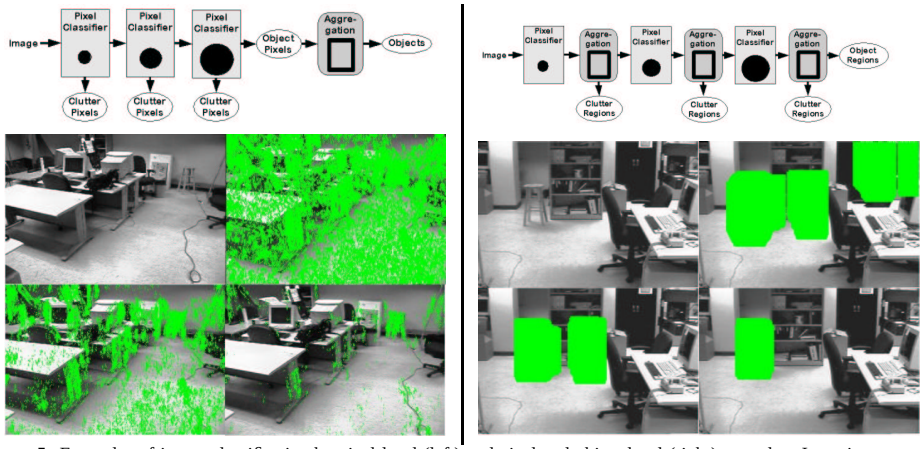


Figure 5: Examples of image classification by pixel-level (left) and pixel-and-object-level (right) cascades. Input images are shown at upper left; successive images (left to right, top to bottom) show in green which pixels are still under consideration after processing by progressive cascade phases.

Cascade	# Phases	Testing On Low Internal Clutter (n=98)				Testing On Higher Internal Clutter (n=193)			
		Pixel TP	Pixel FP	Object TP	Object FP	Pixel TP	Pixel FP	Object TP	Object FP
Pixel	35	0.404	0.040	0.296	0	0.382	0.044	0.207	0
Pixel And Object	**12**	**1.000**	**0.080**	**0.827**	**2.480**	**0.954**	**0.082**	**0.881**	**2.756**

Table 1: Summary of quantitative recognition rates for cascades trained on the stool images. Pixel true positive/false positive rates and object true positive/false positive rates are listed as Pixel TP/FP and Object TP/FP. "n" is the number of test images in each category. # Phases is the number of phases in the trained cascade.

floor and its scale varies by a total of about 15% across all frames. The camera translates and rotates between each view. Eleven times over the course of acquiring 557 images we changed the *world configuration*– that is, we shuffled the clutter objects, moved the stool and camera to different parts of the room, and rotated the stool arbitrarily. We manually labeled the stool in each image. We partitioned the world configurations into low-internal-clutter (LIC, see Figure 4(a)) and high-internal-clutter (HIC, see Figure 4(b)) categories [2]. Overall, these images are challenging since some of the edges on the target object are weak and the internal clutter adds high variability to the appearance of object regions [3]. To insure significant variations between the training and test sets, we partitioned images into a training set containing all 266 images from 4 of the LIC world configurations; and a test set containing all 98 images from the 3 other LIC world configurations along with all 193 images from the 4 HIC configurations. We trained a cascade of pixel-object pairs and summarized performance in terms of the percentage of true positive and false positive pixels detected by the cascade, as well as the object true positive rate and average number of object false positives per test image. After the final cascade phase, we reduce dense clusters of object detections to single, isolated detections through non-maximum suppression over a 50x50 window. We count an image region as a true positive if it overlaps with the true object region by 75% or more.

We compare results from our pixel-and-object-level (POL) approach to a pixel-level (PL) approach whose cascade phases solely consist of pixel-level classifiers built using the feature selection and training methods of Sections 2 and 3 (see Figures 3 and 5 for a

[2] "Internal clutter" refers to clutter which appears through holes in the object, for example between the legs and leg supports of the stool.

[3] We focus in-depth on detection of a single object over hundreds of widely-varying images because we feel that doing so gives more insight into algorithm performance than isolated, possibly accidental, anecdotal results on several objects.

Figure 6: Example recognition result from the hybrid pixel-and-object-level cascade. Left to right: test image, recognition result after one cascade phase, recognition result after 12 cascade phases. Boxes shown in green are still under consideration for containing the stool.

comparison of the two approaches), followed by an aggregation filter that adds the number of detected object pixels over image regions *post facto*. This comparison highlights the utility of training the pixel-level and object-level processes in a unified framework rather than treating the two as isolated modules. Both cascades used cost-based ROC grading criteria based on constant misclassification costs of .01 and .99 for false positives and false negatives respectively [21]. Both cascades used 5 operator-offset pairs per pixel classifier, relative probe centers laid out over circular apertures with $r(\Delta_n) = 3 * n$ pixels, $n = \{1 \cdots 5\}$, and an object rectangle size of 31 by 48 pixels. Training the PL and POL cascades took roughly one day and 12 hours respectively, and both take roughly 1 second to classify a novel 360x240 image.

Quantitative results for the cascades are summarized in Table 1. POL achieves higher pixel true positive rates than PL, and their pixel false positive rates are comparable. For both test image categories, the object true positive rates for POL are much higher than for PL (82% *vs.* 30% and 88% *vs.* 21%), with a reasonable number of false positives per image (2.5 and 2.8 respectively). Also note that POL generalizes well to the more difficult HIC images not seen in the training data, while PL does not.

Example results comparing PL and POL on novel views of world configurations not seen in the training data are shown in Figure 5 for several cascade phases. Intuitively, each phase in POL removes entire regions of the image from further consideration, while the pixel-level cascade gradually discards isolated, individual pixels at each phase. This explains why PL requires many more cascade phases (35 *vs.* 12) to converge to acceptable recognition rates. An additional example result for POL is shown in Figure 6.

The reason for the object false positive rates between 2 and 3 for POL is illustrated in Figure 3, bottom row. Since aggregation scores tend to be high at and near the target object, the aggregation score drops off slowly around its peak at the true object location (Figure 3, bottom row, 3rd column), causing a number of object false positives in the immediate vicinity of the object. More advanced post-processing routines, such as those used to reduce multiple, overlapping face detections [19], should address this issue in the future.

8 Conclusion

We presented a framework for localizing complex-shaped objects in images by identifying individual pixels on the object and assembling the pixels into coherent image regions. By optimizing both types of processes in a single, unified training framework, we are able to detect objects in highly cluttered scenes under arbitrary out-of-image-plane rotation.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[2] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[3] Owen Carmichael. *Discriminative Techniques For The Recognition Of Complex-Shaped Objects*. PhD thesis, Carnegie Mellon University, September 2003.

[4] Owen Carmichael and Martial Hebert. Shape-based recognition of wiry objects. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 2003.

[5] Owen Carmichael and Martial Hebert. Word: Wiry object recognition database. www.cs.cmu.edu/~owenc/word, January 2004. Carnegie Mellon University.

[6] G. Dorko and C. Schmid. Selection of scale invariant neighborhoods for object class recognition. In *Proceedings International Conference On Computer Vision*, 2003.

[7] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2001.

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, volume 2, pages 264–271, 2003.

[9] W.E.L. Grimson. *Object recognition by computer : the role of geometric constraints*. MIT Press, 1990.

[10] Stan Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang, and Harry Shum. Statistical learning of multi-view face detection. In *Proceedings European Conference On Computer Vision*, 2002.

[11] X. Liu, A. Srivastava, and K. Gallivan. Optimal linear representations of images for object recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 2003.

[12] David Lowe. Object recognition from local scale-invariant features. In *Proceedings International Conference On Computer Vision*, pages 1150–1157, 1999.

[13] David G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Boston, Mass., 1985.

[14] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *Proceedings British Machine Vision Conference*, 2003.

[15] T. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1 edition, 1997.

[16] P. Mitra, C.A. Murthy, and S.K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

[17] Stepan Obdrzalek and Jiri Matas. Object recognition using local affine frames on distinguished regions. In Paul L. Rosin and David Marshall, editors, *Proceedings British Machine Vision Conference*, volume 1, pages 113–122. BMVA, 2002.

[18] J.R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[19] Henry Schneiderman and Takeo Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 2002.

[20] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection and recognition. In *Proceedings International Conference On Computer Vision*, pages 18–25, 2003.

[21] Peter Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, Stanford University, California, 2000.

[22] Nuno Vasconcelos. Feature selection by maximum marginal diversity: optimality and implications for visual recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, volume 1, pages 762–769, June 2003.

[23] Paul Viola and Michael Jones. Robust real-time object detection. Technical Report CRL 2001/01, Compaq Cambridge Research Laboratory, 2001.

[24] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *Advances in Neural Information Processing Systems (NIPS*, 2003.

[25] Stella X. Yu and Jianbo Shi. Object-specific figure-ground segregation. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, Madison, Wisconsin, June 2003. IEEE.