Shape-Based Recognition of Wiry Objects

Owen Carmichael, Member, IEEE, and Martial Hebert, Senior Member, IEEE

Abstract—We present an approach to the recognition of complex-shaped objects in cluttered environments based on edge information. We first use example images of a target object in typical environments to train a classifier cascade that determines whether edge pixels in an image belong to an instance of the desired object or the clutter. Presented with a novel image, we use the cascade to discard clutter edge pixels and group the object edge pixels into overall detections of the object. The features used for the edge pixel classification are localized, sparse edge density operations. Experiments validate the effectiveness of the technique for recognition of a set of complex objects in a variety of cluttered indoor scenes under arbitrary out-of-image-plane rotation. Furthermore, our experiments suggest that the technique is robust to variations between training and testing environments and is efficient at runtime.

Index Terms—Object recognition, edge and feature detection, classifier design and evaluation, shape.

1 INTRODUCTION

Over the past 10 years, significant progress has been made toward the recognition of real, complex objects in cluttered scenes. There are now object recognition systems whose detection and false alarm rates are encouraging for real-world applications [42]; some of these promising systems even perform in real time [46]. The most common target object searched for is the human face, but, in principle, these systems could be trained to detect any of a variety of objects including cars and buildings.

Many of these approaches formalize the recognition problem as one of modeling the appearance of a rectangular image patch circumscribing the object, across changes in pose [38], lighting [5], or other conditions. Thus, the recognition problem is reduced to examining a specific rectangular image template, and using its appearance to decide whether or not it is the bounding box around the image of the target object.

Since the problem is formulated in terms of rectangular image windows, appearance-based recognition methods work well when applied to target objects whose projection into the image fills a rectangular region. However, many objects produce images that are poorly approximated by rectangles; for objects such as the ladder, cart, and chair in Fig. 1, their bounding boxes in the image contain a high percentage of pixels that map to the background or other objects. Successful recognition techniques can handle the variation in template appearance induced by a small number of background pixels in the image patch. When most of the template consists of clutter, however, its appearance can vary widely due to a modification of the viewing environment or object pose, making it difficult to detect the object based on the entire template. This paper focuses on the recognition of objects like the ones in Fig. 1; we will refer to them as wiry objects since they consist mainly of elongated, thin stick-like components connected together into complex structures.

An alternative to recognition from a single template, proposed by several authors [7], [16], [28], [43], is to break up the image representation of the object into a collection of smaller rectangles, each of which corresponds to a subsection of the object. This strategy may be effective for some objects; consider, however, wiry objects such as the chair in Fig. 1.

Any image template larger than a few pixels across will intersect mainly clutter pixels when placed over the legs or armrests, and it is doubtful that image patches a few pixels wide will contain sufficient information to discriminate the appearance of the object from the background.

Furthermore, popular approaches to object recognition analyze the greylevel or color texture patterns in candidate patches; thus they tend to work well when the target object has significant visual texture. Faces, cars, buildings, and many other common objects possess this characteristic. But, for the near-monochrome objects in Fig. 1, along with many other common objects, there will be too little appearance variation to use texture as a cue for discrimination. Thus, while template-based techniques are effective for some objects, we feel it is worth investigating the problem of recognition from alternative cues, especially shape. Specifically, we employ discriminative machine learning techniques to boost the effectiveness of the edge-based recognition paradigm popular in the 1980s [22] to the point of feasibility in high-clutter scenes under significant 3D pose variation.

This paper presents an efficient technique for using example images of a particular wiry object, like the specific chair in Fig. 1, in typical environments to automatically train a classifier cascade that localizes the object in highly cluttered novel views under arbitrary out-of-image-plane rotation.1 Given an input image $I$ (Fig. 1, first column), we

1. We note that it is possible to extend our technique to handle significant object scale variations either by processing the same image repeatedly at a variety of scales [42] or by rectifying image features to a canonical scale [34], [20]. We also note that we focus on detecting a single, individual instance of a wiry object across difficult 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 ones.
extract binary edges (Fig. 1, second column) and use the configuration of the edges to determine which edge pixels belong to an instance of a target object, and which edge pixels belong to clutter (Fig. 1, third column). Once the individual edge pixels belonging to the target object have been identified, it is possible to group them into overall hypotheses about the presence of the object (Fig. 1, fourth column); alternatively, the object edge pixels may be provided as input for further recognition procedures such as pose estimation [3], [45]. Section 3.5 describes a simple mechanism for clustering a set of individual object edge pixels into an object-level detection; however, the bulk of this paper is concerned with the preceding problem of object-background separation in edge images, or edge filtering: using shape cues to decide which of the edge pixels project to the target object.

Formally, let $\mathcal{G}$ denote a list of pixels $q = [x, y]$ from a novel image $I$ such that an edge has been detected at $I[q]$. Our goal is to use $\mathcal{G}$ to recover a second list, $O$, which contains only those edge pixels $q \in \mathcal{G}$ which correspond to points on an instance of a particular target object of interest. In more detail, our objective function for this task is to maximize $tp_o$, the number of true positive pixels in $O$; and minimize $fp_o$, the number of false positive pixels in $O$. True positive pixels are edge pixels $q \in O$ that project onto the target object, and false positive pixels are $q \in O$ that project onto the clutter. Object pixels we accidentally fail to include in $O$ we refer to as false negative pixels and background pixels we (correctly) exclude from $O$ are true negative pixels. False negative pixels and false positive pixels are mistakes: They are the object pixels we accidentally classify as background and the background pixels we accidentally classify as object, respectively. In Section 3.4, we discuss the fact that the relative importance of the two kinds of mistakes may vary from problem to problem, and we provide classification techniques based on analysis of receiver operating characteristics to address this issue. Our only source of training data is a set of images containing the target object in typical scenes, from which edges have been extracted and labeled “object” or “clutter.” In other words, at training time we are given a set of images $T = \{T_1, \ldots, T_n\}$ and a set of edge lists, $T = \{T_1, T_2, \ldots, T_n\}$, where each $T_j$ is composed of two sublists $T_{j+}$ and $T_{j-}; T_{j+}$ consists of edge pixels $q_+$ extracted from image $T_j$ that correspond to a point on the target object, while each $q_- \in T_{j-}$ is an edge pixel that maps onto the background environment.

We present a cascade approach to recovering $O$. At each pixel on each edge, we examine the edges in a neighborhood surrounding it, which we call the aperture of the edge pixel (Fig. 3a). How edges are arranged inside the aperture, termed the local edge configuration of the edge pixel, is the cue used to determine whether that edge pixel belongs to the object or the background. A classifier is trained from example views to discriminate local edge configurations of clutter edge pixels from those of object edge pixels. Unfortunately, if the aperture is too small, the local edge configuration may be ambiguous; in other words, it might be impossible to tell which class the edge pixel belongs to based on edge information inside the aperture. For this reason, ambiguous edge pixels are passed on to a second classification phase that considers the local edge configuration in a larger aperture. If it is still unclear at this stage whether the edge pixel belongs to the foreground or background, we attempt to classify it based on the edge configuration in a larger aperture, etc. As an illustration, Fig. 2 depicts four phases in this cascade process for the recognition of the chair in the lower left portion of the image.

At each phase in the cascade, a discriminative classifier computes a sparse set of localized edge features that measure edge density in a small image neighborhood. The locations of the edge features are learned at training time.

Section 2 discusses related approaches to edge-based recognition. Section 3 describes our algorithm in general. In Section 4, we present a set of experiments that demonstrate
the viability of the technique for the recognition of three everyday wiry objects. Additionally, Section 4 addresses concerns related to the sensitivity and complexity of our algorithm. Section 5 discusses what image features our algorithm learns during the training process. Conclusions and directions for future research are discussed in Section 6.

2 RELATED WORK

Object recognition research in the 1980s culminated in systems that could detect occluded, nonconvex shapes from binary edge images [22]. Interpretation trees [22], for example, use a tree search to explore the space of all possible correspondences between features on an object model and features in the image. Unfortunately, as the number of model features and image features grows, the space of correspondences can grow intractably large, especially if the image contains significant clutter or noise.

Indexing techniques such as geometric hashing [29] avoid the computational load of exhaustive correspondence search, but they too suffer in the presence of clutter and noisy edges. In these approaches, each k-tuple of image features casts votes for the identities and/or poses of objects in the image, based on their geometric arrangement. If the image contains significant noise [23] or clutter, the votes cast by sets of clutter features will overwhelm the votes cast by the object, making it difficult to draw any conclusions about what objects are there. In a broad sense, our approach bears resemblance to indexing methods to the extent that it is bottom-up; that is, we first compute local image features and then assemble overall object detections by reasoning about the properties of those features.

Early indexing techniques computed very simple descriptors from sets of three, four, or five image features; more recently, several authors have proposed the use of richer descriptions of local image shape in conjunction with indexing for edge-based recognition. Conspicuous groupings of edge segments [4], contours in a normalized rectangular image patch [43], and histograms of local edge distributions [6], [36] are all examples of more advanced local edge features that have been applied to shape-based recognition problems. We feel that rich local shape descriptors such as these enhance the discriminative power and efficiency of edge-based recognition. In each of these approaches, however, various parameters controlling the characteristics of the edge descriptor (histogram bin sizes, grouping thresholds, etc.) must be provided by the user; the goal of our approach is to use training examples of the object in typical environments to automatically estimate feature parameters in such a way that the edge properties computed on a test image effectively discriminate the target object from the background.

The closest approaches to ours in terms of problem setting, image features, and classification techniques are due to Mikolajczyk et al. [36], Belongie et al. [6], and Amit et al. [2]. The problem of recognizing wiry objects (specifically, bicycles and badminton racquets) in clutter is directly addressed in [36]. There, scale-invariant interest regions are extracted from the image and the orientations and locations
of edges are accumulated into 3D histograms along the lines of the SIFT descriptor [34]. Here, we focus on learning the characteristics of discriminating edge-based features directly from training examples of the object in clutter. We do so to avoid the hand-tuning of histogram parameters (e.g., number of bins and bin extents) required in [36].

Our spatial arrangement of local shape features are inspired by the shape contexts of Belongie et al. [6]. At each edge pixel in an image, a histogram, or “shape context,” is calculated; each bin in the histogram counts the number of edge pixels in a neighborhood near the pixel. Nearest-neighbor search and histogram distance measures then determine correspondences between shape contexts from a test image and shape contexts from model images. Our approach is closely related; both use the distribution of edges in an aperture surrounding a pixel as the fundamental feature for recognition. However, the shape context uses a “dense” set of edge features for recognition; in other words, the bins in the histogram exhaustively cover the entire aperture. Since we only compute edge features at isolated image locations deemed likely to discriminate the edge pixel in question as object or clutter, the features we use are spatially “sparse.” While dense features may be effective when the background is not a concern, we feel that they will represent local edge configurations poorly for the target objects and scenes we consider. Specifically, if the neighborhood surrounding an object edge contains a significant number of clutter edges—consider the objects in Fig. 1—then, many of the shape context bins will be filled solely with background edge pixels which can confound the matching process. This intuition is supported by a recent study assessing shape context matching in high clutter [44]. This is in large part the same reason why rectangular templates represent local appearance poorly on wiry objects: Much of the local object representation will actually consist of image data drawn from clutter.

Our strategy of using decision trees to select and classify edge-based features is close to that of Amit et al. [2]. In their approach, sets of decision trees test whether binary test images contain local features (“tags”) in particular geometric arrangements. Here, we take advantage of some of the same desirable qualities of decision trees: In particular, automatic feature selection and fast runtime performance through sparse sets of local feature computations. The chief difference in our technique is that decision trees are trained in a cascade format for increased training and runtime efficiency. Also, while the tags of [2] measure the configuration of edges over small (e.g., 4 x 4) pixel blocks, our local edge features give an overall measure of edge density over potentially larger neighborhoods.

3 Approach
Our approach to edge filtering consists of training a cascade of classifiers that discriminates object edge pixels from background edge pixels based on local edge density features. Section 3.1 defines the features used to train the classifiers, and Section 3.2 describes how classifiers trained from these features are applied to the image in a cascade format. Sections 3.3 and 3.4 discuss the classifiers themselves.

3.1 Image Features: Edge Probes
As stated above, we assume that our objects are well-characterized by their shape properties, and that the shape properties are represented by binary edges extracted from the image. For this reason, we classify each query edge pixel based on the values of edge features that summarize the local shape in image neighborhoods in its immediate vicinity.

Specifically, an edge probe at probe center \( p \) over a list of edge pixels \( G \) is defined as

\[
ep(p, G) = \sum_{t \in G} \exp \left( -\frac{||t - t||^2}{\sigma^2} \right),
\]

where \( t \) and \( p \) are 2-vectors of \( [x, y] \) image coordinates. An edge probe can be thought of as a Gaussian receptive field with variance \( \sigma^2 \), centered at pixel \( p \) in an edge image whose edge pixels are contained in \( G \). Edge probes measure the density of edge pixels in some neighborhood in the image; in this sense, each edge probe is analogous to a bin in a shape context histogram [6]. The variance \( \sigma^2 \) is a user-set parameter; however, an experiment reported in Section 4.3 suggests that our edge filtering results are not highly sensitive to its setting. See Fig. 4 for an illustration of an edge probe evaluated at probe center \( q + \delta \).

Edge probes summarize the local shape in image neighborhoods in terms of the density of edges in those neighborhoods. By computing edge probes at a set of image neighborhoods surrounding a query edge pixel \( q \), we summarize the overall shape properties of the image near \( q \), which should be a discriminating recognition cue for our shape-based objects.

Specifically, for each query edge pixel \( q \), we compute edge probes at probe centers in the spatial neighborhood of \( q \) to classify it as object or clutter. Consider a set of relative probe centers \( \Delta = \{ \delta_1, \delta_2, \ldots, \delta_k \} \), \( \delta_i = [x_i, y_i] \), laid out over a 2D grid centered at the origin. To classify \( q \), we shift the relative probe centers so that they surround \( q \), and compute edge probes \( \exp(q, \Delta, G) = \exp(q + \delta_1, G), \exp(q + \delta_2, G), \ldots, \exp(q + \delta_k, G) \) at shifted probe centers \( \{ q + \delta_1, \ldots, q + \delta_k \} \). An illustration is shown in Figs. 3a and 3b.

Given a fixed \( \sigma \), we space the relative probe centers at intervals of \( \sigma \) pixels so that they evenly blanket a circular aperture as in Fig. 3a. We arrange the probe centers this way so that every pixel in the aperture is able to contribute to edge probes evaluated at one or more relative probe centers in a local neighborhood. But, how large should the aperture be? We want the shifted probe centers to cover a large enough neighborhood that the edge probes will contain sufficient information to discriminate object pixels from clutter pixels. At the same time, however, if the aperture is too large, an intractable amount of computation will be required at training time to evaluate edge probes that might not be crucial for classification. Worse, if the aperture is so large that most of the edge probes are totally irrelevant to the category of the query edge pixel, error-prone classifiers could be trained [25], [1]. Thus, we are presented with “the aperture problem” that appears in many computer vision problems—when attempting to induce information about a particular location in the image we want to incorporate image data from a large enough surrounding area, but not so large that we introduce irrelevant data or useless computation.\(^2\)

\(^2\) We emphasize that there is a critical distinction between the aperture of an edge pixel (denoted by the black circle in Fig. 3a) and the spatial support of a single edge probe (denoted by the gradient-shaded region surrounding \( q + \delta \) in Fig. 3b)—the aperture describes the image region over which all edge features for a given query pixel are evaluated, while the portion of the image that contributes to a single edge feature is determined by the edge probe support.
3.2 The Cascade

We solve our aperture problem in phases. We first identify those edge pixels whose class is discriminable based on very nearby features, then identify edge pixels that are made discriminable by features slightly farther away, and continue to do so until the aperture covers the entire object in question. The overall objective for the cascade is to maximize the number of true positive pixels and minimize the number
of false positive pixels. As an illustration, Fig. 2 shows the results of classification at four phases of the cascade.

In addition to providing a solution to our aperture problem, the classifier cascade allows fast screening of edge pixels whose classification is easily determined based on information in a small window, leaving the bulk of the computation to more ambiguous sections of the image. Similar cascades have recently achieved significant speed-ups for template-based approaches to recognition—see, for example, [46], [31], [24], [47], [32].

More formally, consider a set of relative probe centers Δ that cover a circular aperture as in Fig. 3a. Define r(Δ) to be the radius of the circle. Our approach is to train a series of one-sided cascades for two reasons. First, they are simpler to train because they involve classifying edge pixels into clutter edge pixels (which are permanently discarded), as well as high-confidence object edge pixels (which are permanently classified as “object”), and pass the remaining, less-certain edge pixels to subsequent phases (see [32] for example). Two-sided cascades have the computational advantage of removing more edge pixels from classification earlier on since they remove both object and clutter edge pixels from consideration at each phase. However, we build one-sided cascades for two reasons. First, they are simpler to train because they involve classifying edge pixels into two categories instead of three. Second, we assume that in our highly cluttered images the object edge pixels are relatively “rare events” [47], meaning that removing object edge pixels from the cascade early on will only yield slight computational savings.

### 3.3 Decision Trees

We seek classifiers \{c_1, c_2, \ldots, c_k\} that compute sparse sets of edge probes over apertures \{Δ_1, Δ_2, \ldots, Δ_k\}. The classifiers assign each of the edge pixels in a set of training images to object or background categories; the objective of the classifiers is to minimize the number of false positive pixels and maximize the number of true positive pixels, among the training images. Our assumption is that classifiers that do so will achieve high true positive rates and low false positive rates when applied to novel views of the target object and its environment.

We choose decision tree classifiers [41] for their ability to quickly classify based on small, sparse sets of features. To classify a query edge pixel \(q \in G\), we evaluate edge probes at the probe centers associated with tree nodes, and threshold the edge probe values to determine how to traverse the tree from the root node to a leaf. Associated with each leaf is a score \(w\) that represents the probability that the query edge pixel belongs to the object. By thresholding \(w\), we arrive at a binary decision about whether the edge pixel corresponds to the object or background.

Our training procedure for decision trees is a two-step process of tree generation and pruning, following the reduced-error pruning approach of Quinlan [41]. In this framework, the training data is split into two subsets, which we will refer to as the tree-growing set and the holdout set. We use standard tree induction techniques to build a decision tree with high classification accuracy on the tree-growing set; the score associated with each leaf is estimated by counting the number of object edge pixels and background edge pixels classified to the leaf. Then, subtrees are pruned from the tree when doing so improves a performance criterion on the holdout set [41], [9], [11].

Section 3.4 discusses pruning. Tree induction, in more detail, works as follows: We begin by considering all possible binary splits \((t, δ_t)\) of the training set into two categories \(A\) and \(B\) such that \(ep(q_+, δ_t, G) < ep(q_+, δ_t, G)\) \& \(q_+, q_+ \in A, q_+ \in B\). We employ the widely used information gain criterion [37] to assign scores to all splits based on how well they partition the training data into categories containing high percentages of object and background pixels respectively (see [27], [12] for alternative criteria). The tree-growing set is recursively split until no high-scoring splits are available.

Each leaf of the tree corresponds to a set \(S\) of training examples, composed of a subset \(S_o\) of object edge pixels and a subset \(S_g\) of background edge pixels. The score we assign to each leaf is its Laplace-corrected empirical probability \(|S_o| + (c - 1))/(|S| + c); this score is a smoothed estimate of the probability that an edge pixel classified to that leaf corresponds to the target object. We chose this score following [9], which showed that for some problems using Laplace-corrected scores led to higher classification performance than the empirical probability \(|S_o|)/(|S|). Here, we choose \(c = 2\) as in [9]; however, we note that it is possible to tune \(c\) for optimal performance as in [48].

Rather than assign real-valued scores, it is more traditional for the tree induction procedure to have each leaf assign a binary classification to the edge pixels. For example, early approaches to decision tree induction labeled all edge pixels classified to a leaf as “object” if \(|S_o| + (c - 1))/(|S| + c) > .5\. We assign real-valued scores to allow for flexibility at pruning time. Specifically, our pruning procedure described in Section 3.4 simultaneously removes subtrees from the decision tree and selects a threshold \(t\) such that all leaves with scores \(|S_o| + (c - 1))/(|S| + c) > t\ classify edge pixels as “object.” By modulating \(t\) and changing the topology of the tree, the pruning procedure has greater flexibility in optimizing performance criteria than if it were only allowed to change the tree topology. In other words, by postponing binary decisions at leaves until pruning, we allow the pruning procedure to optimize the tree to greater performance.

We chose decision trees based primarily on two desirable qualities. The first is sparsity; decision trees are capable of classifying examples based on a sparse set of features that are selected by the training algorithm at each node. In practice (see Figs. 9 and 10 for example), we find that in most cases a small subset of the total number of possible edge probes are evaluated in order to classify each pixel.
The second desirable property is that the framework for training decision trees consists of a discrete optimization framework of adding and deleting sets of nodes. That is, we can phrase the problem of tuning the decision tree in terms of adding or deleting nodes if doing so increases the criterion. In the following section, we describe how we tune grown decision trees so that they increase a performance criterion related to the relative importance of false positive pixels and false negative pixels.

3.4 Pruning

As described above, our goal is to train the decision tree classifier to produce high rates of true positive pixels and low rates of false positive pixels. However, as we describe below, optimizing the decision tree often involves a trade-off between false positives and false negatives. In this section, we describe how to represent this trade-off mathematically as a receiver operating characteristic (ROC) curve, and how to tune decision trees based on the characteristics of ROC curves.

3.4.1 ROC Curves

At run time, the decision tree processes each pixel \( q \) in the test image and assigns it a real-valued score \( w \). We arrive at a binary classification for the pixel by thresholding \( w \)—if \( w \) is higher than a threshold \( t \), \( q \) is classified as “object,” and if not it is classified “background.” Some number of object edge pixels in the test image will have classification scores higher than \( t \); these are the true positive pixels for that classifier and threshold setting. Object edge pixels with scores below the threshold are the false negative pixels since they are falsely classified as background. Background edge pixels with scores higher than the threshold are the false positive pixels; background edge pixels with scores below the threshold are the true negative pixels. Usually, setting \( t \) involves a trade-off between false positives and false negatives; for example, if \( t \) is set very high, the classifier will be very “selective” so that very few background pixels will have scores \( w > t \), meaning that there will be few false positives. However, if \( t \) is very high there will be many object pixels with scores \( w < t \), meaning the false negative rate will be high.

The ROC curve provides a way to visualize the trade-off between false positives and false negatives by plotting the number of true positive pixels \( tp \), as a function of the number of false positive pixels \( fp \), for all possible settings of \( t \). See Fig. 4 for an example. Each point \((fp, tp)\) on the curve is referred to as an operating point. ROC curves have historically been used to represent this trade-off in signal processing systems (see, for example, [18]) and, more recently, the machine learning community has used them to analyze how classifiers perform over a range of threshold settings [40], [10]. Here, we incorporate ROC curves into decision tree pruning criteria during training.

3.4.2 ROC-Based Pruning

Since ROC curves represent the trade-off between false positives and false negatives over a range of decision threshold settings, our pruning procedure aims to remove subtrees from the decision tree whenever doing so improves a performance criterion called an ROC grading criterion assessed on the ROC curve. The ROC grading criterion reflects the relative importance of false positives and false negatives, which may vary from application to application. For this reason, our pruning framework is general enough to incorporate arbitrary ROC grading criteria.

Specifically, during pruning, we look at each subtree in turn and consider two different classifiers: \( c_1 \) consists of the full decision tree, and \( c_2 \) consists of a modified tree in which the current subtree has been replaced by a leaf node. We classify the holdout images with each classifier and graph the trade-off between false positives and false negatives for the two classifiers as ROC curves \( r_1 \) and \( r_2 \). We represent the ROC curves as functions mapping \( fp \) to \( tp \). The ROC curves are then evaluated using an ROC grading criterion, \( grade \), which assigns a numerical score \( grade(r_1) \) and \( grade(r_2) \) to \( r_1 \) and \( r_2 \). If \( grade(r_2) > grade(r_1) \), we remove the subtree from the tree; otherwise, the tree stays unaltered. After pruning, we invoke a threshold function \( t = h(r) \) that returns a decision threshold (or, equivalently, an operating point) for the classifier. Our inductive bias is that the ROC curve for the holdout pixels will generalize well to novel test images, so that a particular threshold \( t \) will lead to similar operating points on the holdout and test images.

As stated above, the ROC grading criterion and the threshold function reflect the performance priorities of the application at hand in terms of false positives and false negatives. Here, since the classifiers we train are modules in an overall cascade, our ROC grading criterion encourages the classifiers to contribute to low false negative and false positive rates for the cascade as a whole.

In particular, consider Figs. 4e and 4f, corresponding to the green and red operating points in Fig. 4d, respectively. In the former case, the background pixels classified as “object” will be passed to further classifiers in the cascade, which may in turn reclassify these pixels correctly. In the latter case, object pixels classified as “background” will be discarded by the classifier and will not be reconsidered by later classifiers. In other words, false positives are recoverable errors and false negatives are irrecoverable, so false negatives incur a much higher penalty than false positives. For this reason, our threshold function selects operating points to encourage the lowest \( fp \) possible while constraining \( tp \) to a fixed, high percentage \( \theta \) of the number of object edge pixels. In so doing, we push the classifier to make progress on the overall cascade goal of having few false positives while constraining it to contribute as few errors as possible—false negatives—that later cascade phases are unable to recover from. This strategy can be thought of as the converse of a Neyman-Pearson criterion [17], which attempts to maximize \( tp \) given a fixed threshold on \( fp \). Furthermore, our ROC grading criterion is based on the same fixed detection rate criterion; that is, if \((fp, tp)\) is an operating point for our ROC curve, then the grading criterion is inversely proportional to \( fp \).

Setting the parameter \( \theta \) requires assumptions about the expected number of phases in the cascade and the expected difficulty in separating object edge pixels from background. On one hand, setting \( \theta < 1 \) allows the classifier to effectively “give up” on \( 1 - \theta \) percent of the most difficult object edge pixels in the holdout set, which gives the classifier a certain amount of robustness to outliers. On the other hand, if the each classifier “gives up” on \( 1 - \theta \) percent of the object edge pixels in its training set, then after \( k \) cascade phases the true positive rate of the overall cascade will be \( \theta^k \). Good
values for $\theta$ require balancing the need to discard difficult
object edge pixels as false negatives, while trying not to
degrade the overall cascade true positive rate too much.

The training and runtime behavior of our edge-filtering
algorithm are summarized in Algorithm 1 and Algorithm 2.
Here, for a given set of edge lists $G = \{G_1, \ldots, G_k\}$, $c(G) = \{c(G_1), \ldots, c(G_k)\}$, and
\[
c(G_i) = \{ q \in G_i \mid c \text{ classifies } q \text{ as 'object'} \}.
\]
$G_{\text{c}}$ is the subset of $G$ containing object edge pixels extracted
from the image and $G_{\text{c}}$ contains clutter edge pixels.

Algorithm 1 Pseudocode for the training procedure for
recognition based on a cascade of edge probes.

Require: Edge lists $G = \{G\}$, sets of probe centers $\{\Delta\}$, $\theta$, $\sigma$.
1: Split $G$ into a tree-growing set $T$ and a holdout set $H$.
2: Let $FP = \{q \in T \cup H\}$, $TP = \{q \in T \cup H\}$.
3: for all $\Delta$ do  // loop over cascade phases
4:   for all $T_i \in T$ do  // loop over the tree-growing set
5:     for all $q \in T_i$ do  // loop over object edge pixels
6:       $z_{q_i} = ep(q_i, \Delta, T_i)$
7:     end for
8:   for all $q \in T$ do  // loop over clutter edge pixels
9:     $z_{q_i} = ep(q_i, \Delta, T_i)$
10: end for
11: end for
12: Train a decision tree $c$ to discriminate $\{z_{q_i}\}$ from $\{z_{q_i}\}$
13: for all $H_i \in H$ do  // loop over the holdout set
14:     for all $q \in H_i$ do  // loop over object edge pixels
15:       $z_{q_i} = ep(q_i, \Delta, H_i)$
16:   end for
17: for all $q \in H$ do  // loop over clutter edge pixels
18:     $z_{q_i} = ep(q_i, \Delta, H_i)$
19: end for
20: end for
21: Prune $c$ to minimize $|c(H_\text{c})|$ such that
\[
|c(H_\text{c})| = (1-\theta) \times |H_\text{c}|.
\]
22: if $|c(H_\text{c})| < |H_\text{c}|$ then
23:     Add $c$ to cascade $C$.
24: $H = c(H), T = c(T)$  // discard correctly classified
background edge pixels
25: end if
26: end for

Algorithm 2 Pseudocode for the runtime evaluation of a
novel image based on a cascade of edge probes.

Require: List of edge pixels $G$
1: $O = G$
2: for all $c \in C$ do  // loop over classifiers in cascade
3:   $O = c(O)$  // discard background edge pixels
4: end for
5: Return $O$

For each edge pixel $q$ in each image in the tree-growing
set, we compute edge probes at all shifted probe centers $\{q + \delta_1, \ldots, q + \delta_i\}$ corresponding to the relative probe
centers $\{\delta_1, \ldots, \delta_i\}$ in the smallest aperture $\Delta_1$. Decision
tree induction then iteratively splits the edge pixels into
subsets according to the values of edge probes corresponding
to selected relative probe centers. Each edge pixel in the
holdout set images is then classified by the resulting tree,
and subtrees are removed if the pruned tree reduces the
number of background edge pixels classified as object edge
pixels while keeping the percentage of object edge pixels
correctly classified to $1-\theta$. Edge pixels from the tree-
growing and holdout sets classified as object edge pixels by
the pruned tree then pass to the training of the second
phase in the classifier cascade: For each of these edge pixels,
edge probes are computed at all shifted probe centers corresponding
to relative probe centers in $\Delta_2$, etc.

Given a test image, we apply the trained classifiers to each
of its edge pixels in turn. An edge pixel classified as “object”
by the first classifier is passed to the second classifier; the
second classifier classifies the pixel again, etc., until the pixel
is labeled as “clutter” or the cascade ends.

3.5 Aggregation
To give an example of how filtered edge images may be
used as an input to aggregation processes, we implemented
a simple filter called an aggregation filter that scans the
filtered edge image with a rectangular template roughly the
size of the target object. At each image location, the number of
degree pixels falling inside the template is recorded, and
image locations with a large number of edge pixels inside
the template are noted as likely locations of the target object
(see Fig. 5).

The justification for this aggregation scheme is its
simplicity at training and runtime. While aggregation filters
do not reason about how the pixel scores are distributed spatially (as in standard template-based approaches to
recognition, for example [42]), they are extremely fast to
apply at runtime since they sum over regions of the filtered
edge image. Furthermore, “training” an aggregation filter
consists of estimating a single parameter: an acceptance
threshold $t$ on the number of pixels inside the rectangle.

We first generate a rectangular template by computing
the bounding boxes of object edge pixels in each training
image, and averaging the size of those bounding boxes. This
gives us a characteristic bounding box $b_I = [b_{x_I}, b_{y_I}]$ whose
width $b_{x_I}$ and height $b_{y_I}$ represent the mean size of the
object in the training images. Next, we run each training
image through the cascade of edge probes described above.
For each training image, consider a binary image $B$ such
that $B[q] = 1$ if pixel $q$ is classified as belonging to the object,
and $B[q] = 0$ otherwise. We apply an aggregation filter to
each location in these binary images:
\[
ag(b_I, q, B) = \sum_{x \in [-\frac{b_{x_I}}{2}, \frac{b_{x_I}}{2}]} \sum_{y \in [-\frac{b_{y_I}}{2}, \frac{b_{y_I}}{2}]} B[q+x, q+y].
\]

If $ag(b_I, q, B)$ is high, the aggregation step reports the
presence of an object instance centered at $q$ and covering a
box of width $b_{x_I}$ and height $b_{y_I}$ in the corresponding
training image. The set of aggregation scores for target
object and background portions of the training images give
rise to another ROC curve, this one used to determine a
threshold on $agl(b,q,B)$ so that a high number of true positive boxes are reported—that is, image locations $q$ such that a bounding box centered there with size $[b_{w1}, b_{w0}]$ covers most or all of the target object—with a minimum of false positive boxes. Figs. 1 and 8 show test images that used an aggregation filter whose threshold was set by hand.

We emphasize that this particular aggregation procedure is in no way an optimal procedure for recovering an object-level description of the contents of the image; in particular, a variety of alignment algorithms [3], [45] would be able to give a more precise correspondence between the test image and a reference image or 3D model. Still, this experiment illustrates that the edge filtering procedure can be a useful preprocessing step to higher-level, edge-based recognition processes that may fail in extreme clutter. For example, Fig. 6 shows a test image for which the aggregation filter fails when applied to the raw edge image, but is able to localize the object once the clutter edges have been removed by the classifier cascade.

4 EXPERIMENTS

To validate our approach, we address the problem of detecting three common objects in highly cluttered indoor scenes under high variation of out-of-image-plane rotation. All images are drawn from the freely available Wiry Object Recognition Database (WORD) [15]. We evaluate the performance of our edge filtering procedure by computing the true positive rate and false positive rate in each image. We emphasize that since we represent objects at a pixel level, “true positive rate” does not mean “percentage of times the object was detected.” Instead, it means “percentage of object pixels detected.” Thus, even if the true positive rate is below 100 percent, it may still be possible to conclusively locate the object in all test images since the density of object edge pixels will be relatively high at the true location of the object if the true positive rate is relatively high. Likewise, “false positive rate” does not mean “number of times a section of the background was mistakenly labeled as the object,” but rather to “number of times an edge pixel in the background was mistakenly labeled as belonging to the object.” Thus, even if the false positive rate is greater than zero, it may be possible to achieve zero false detections of the object, especially if the falsely detected background edge pixels are sparsely distributed in the scene. Indeed, as explained in Section 3.5 and indicated in Figs. 1 and 8, the filtered edge image is an encouraging starting point for estimating the overall location of the object in the image.
Fig. 7. Example foreground and background images and CROC plots for recognition of the cart and chair. See Section 4.1. (a) Foreground images. (b) Background images. (c) CROC plot for the chair. (d) CROC plot for the cart.

### 4.1 Chair and Cart

For our first set of experiments, we took 150 images each of the cart and chair against a blue screen (Fig. 7a). The images span the full revolution of the objects in the plane parallel to the floor. The elevation of the camera varies by approximately 25 degrees with respect to the object, and the extent of scale variation across images is about 10 percent. Note that for some of our images, the generic viewpoint assumption is violated [21]; for example, two of the legs of the chair in Fig. 7a, second row, fourth column, are accidentally aligned.

We also took images of a background environment consisting of a set of “office” objects—for example, lamps, a table, and boxes (Fig. 7b). The set of views spans roughly 60 degrees of rotation in the plane parallel to the floor, and variation in scale and camera elevation is about the same as for the cart and chair images. To induce appearance variation in the background between views, we modified the poses of each background object and shuffled their relative positions every five to eight images. The camera was moved between each view.

The images used as training and testing data for the chair and cart are composites of random pairs of foreground and background images (Fig. 1, bottom row, and Fig. 2 show examples). The pairing was done without replacement; in other words, there is no repetition of foreground or background images in the composite images.

The relative probe centers were arranged as a set of concentric rings; specifically, the relative probe centers in the \( n \)th ring were positioned in a circle of distance \( n \sigma \) from the origin, with a \( \sigma \)-pixel spacing between adjacent relative probe centers on the circle. The set of relative probe centers \( \Delta_n \) corresponding to the \( n \)th aperture in the cascade is the union of all relative probe centers in rings 1 through \( n \) (Fig. 3a). Note that uniformly tiling the aperture with a set of edge features of equal spatial support is in contrast to techniques that aim for a “foveal” layout of edge features, for example [6].

To train the classifiers, we first discretize the edge probe values using the implementation of minimum-entropy discretization [19] in the MLC++ software library [26]; decision trees were grown and pruned as described in Sections 3.3 and 3.4. We set our target false negative rate \( \theta \) to 2 percent.

For each recognition trial on the chair and cart, 150 composite images were partitioned into a tree-growing set of 50 images, a holdout set of 50 images, and a test set of 40-50 images. Binary edges were detected on all images using the Vista software package [39]; specifically, we detected Canny edges and postprocessed them using a procedure (VSegEdgesIntoLines) that fits line segments to the edge pixels and discards short, weak line segments (see [35], [33] for details). Detected edges and edge detection parameters for all images used in our experiments are part of the WORD data set. For computational reasons, we sample the detected edges at five pixel intervals and classify the edge samples. In each image, the ratio of the number of clutter edge pixels to foreground edge pixels is approximately 10 : 1.

A sample result on the chair is shown in Fig. 2; the images show classification results after the 1st, 5th, 10th, and 15th cascade phase. Note that as successive classifiers are applied, using larger apertures, the number of background edge pixels is reduced dramatically while retaining a high number of edge pixels on the chair. Sample input images, detected edges, and filtered edges for the cart and chair are shown in Fig. 1. Note that, in both examples, false positives are so sparse and isolated after edge filtering that they are easily removed by a density filtering mechanism as described in Section 3.5 and shown in the fourth column of the figure.

We visualize the performance of a cascade of classifiers quantitatively using cascade ROC, or CROC, plots. We classify all query edge pixels \( q \) in the test images with the first classifier in the cascade and evaluate the performance of the first classifier via its true positive rate \( tp_1 \) and false positive rate \( fp_1 \). The edge pixels classified as “object” are passed to the second classifier in the cascade, which classifies them with some true positive and false positive rates \( tp_2 \) and \( fp_2 \), etc. A CROC plot for the cascade plots \( \{ (tp_1,fp_1), (tp_2,fp_2), \ldots, (tp_k,fp_k) \} \) for each of the classifiers in the cascade. CROC plots succinctly summarize the contribution made by each of the cascade phases toward the final goal of discarding all false positives while retaining all true positives. A label \( F_n \) next to a point in a CROC plot represents that the point corresponds to the \( n \)th classifier in the sequence.
The performance of each tree in the cascade, over all test images containing the chair, for \( \sigma = 10 \) pixels, is summarized in the CROC plot in Fig. 7c. We performed seven recognition trials; each trial consisted of randomly partitioning the images into tree-growing, holdout, and test sets, training the cascade, and evaluating the true positive and false positive rates of the cascade as more phases are added. Thus, the point marked \( F_1 \) plots true positive rate as a function of false positive rate for a cascade consisting of one classifier, \( f_1 \); \( F_5 \) plots the performance of a cascade containing \( f_1, f_2, \ldots, f_5 \); etc. More specifically, for each cascade, we compute the true positive rates \( TP = \{ n_{ij} \} \) and false positive rates \( FP = \{ p_{ij} \} \) for each test image \( i \) and recognition trial \( j \). The “x” in the graph plots \( (\text{mean}(FP), \text{mean}(TP)) \); bars extend to the left and right by \( \text{var}(FP) \) and up and down by \( \text{var}(TP) \). Fig. 7d shows an analogous graph for results of six recognition trials with the cart. For both objects, the results for \( \sigma = 5 \) pixels and \( \sigma = 20 \) pixels are similar. The true positive and false positive rates for the two objects are comparable—for example, roughly 70 percent of edge pixels on the object are retained, versus 5 percent false positives among the background.
4.2 Ladder

For a second set of experiments, we took 1,157 1,600-by-1,200 images of a ladder in seven different indoor environments: a classroom, conference room, office, lab, living room, warehouse, and kitchen (Figs. 1 and 8). We emphasize that these images are plain photographs of scenes, with no compositing involved. For each image, the camera was approximately 3 m away from the objects in the scene; the elevation of the camera varied between 1.6 m and 1.75 m; the set of all images of a particular scene covered about 60 degrees of rotation with respect to the scene objects in the plane parallel to the floor. The camera was moved between each view, and once every five views the ladder was rotated to an arbitrary angle with respect to the ground and the poses and configurations of clutter objects were randomly modified. The depth of the ladder with respect to the camera varied by a total of approximately 20 percent across all views. Edges were detected in these images using Vista, and edges were hand-labeled as belonging to the ladder or to the clutter objects.

For each experiment, we selected images for tree-growing and holdout sets, and trained cascades of classifiers to filter out background edge pixels as described above. For the experiments in Sections 4.2.1 and 4.2.2, the edge probe variance parameter σ was set to 20 pixels; the decision tree pruning parameter B was set to 2 percent. Depending on the arrangement of edges in the training images, it is possible that the decision tree trained for a particular cascade phase may not significantly reduce the number of false positives on the holdout set; therefore, for our experiments on the ladder we skipped a cascade phase if it failed to reduce the false positive rate by 5 percent or more.

4.2.1 Training and Testing in a Single Environment

First, we considered training individual classifier cascades for each indoor environment separately. For each environment, we randomly split the set of all images of the object in that scene into a tree-growing set of 60 images, a holdout set of 60 images, and a test set containing the remainder of the images. Then, the procedure described in Section 3 was used to train a 20-phase classifier cascade and run each of the test images through the resulting edge filter. For each test image, we measured the true positive and false positive rates after the final cascade phase. Results are summarized in Table 1; examples are shown in Figs. 1 and 8. As in the experiments on the chair and cart, each classifier cascade retained a high percentage of edge pixels on the object (roughly 71-78 percent), while discarding most background edge pixels (roughly 90 percent). Examples of aggregating filtered edge pixels using the approach of Section 3.5 are shown in the rightmost columns of Figs. 1 and 8.

4.2.2 Distinct Training and Testing Environments

Next, we address the address the training of a classifier cascade across a particular set of viewing environments and applying the resulting edge filter to images of the same object in front of entirely distinct clutter. To suggest that the performance of our classifier cascades degrades gracefully according to the deviation between training image characteristics and test image characteristics, we trained a classifier cascade on a set of images of the ladder in five of the rooms (kitchen, cubicle, warehouse, living room, and lab), and tested it on images of the other two rooms (classroom and conference room). We randomly selected a total of 120 images from the set of all images of the object in the training environments, using 60 of them for the tree-growing set and 60 for the holdout set. A classifier cascade was computed from these training images and applied to all images of the object in the test environments, i.e., environments not present in the training data. Edge pixel classification results are summarized in Table 2, using the same notation as Table 1. Comparing the corresponding lines in Tables 1 and 2, it appears that filtering performance decreases slightly in some aspects: The true positive rate on the classroom images drops somewhat when the classroom images are not present in the training data, and the false positive rate for the conference room images increases when the conference room images are absent from training. However, some decrease in performance is to be expected when conditions in training and test images vary significantly; the key point is that the experiment suggests that edge filtering performance degrades gracefully with these variations.

4.2.3 Sensitivity

The edge probe variance parameter σ is a free parameter that critically affects the size of the spatial support region for our basic edge features. Thus, it is natural to wonder how the performance of our approach depends on the choice of σ. To address this issue, we trained a set of four classifier cascades on the images of the ladder in the
confidential room (Fig. 8, fourth row), corresponding to \( \sigma \) values of 15, 20, 25, and 30 pixels. As in Sections 4.2.1 and 4.2.2, the relative probe centers were arranged in concentric rings at distances \( \sigma, 2 \sigma, \cdots \) pixels from the origin. In the cascades for \( \sigma = \{20, 25, 30\} \), a single ring of relative probe centers was added to the aperture at each phase, meaning that the distance from the query edge pixel to the furthest relative probe center in the aperture was \( \sigma \) for the first cascade phase, \( 2\sigma \) for the second cascade phase, etc. In the cascade for \( \sigma = 15 \), we added two rings of relative probe centers per cascade phase, so that, as in the \( \sigma = 30 \) cascade, the distance from the query edge pixel to the furthest relative probe center was 30 for the first phase, 60 for the second phase, etc. 60 images of the confidential room were randomly selected as a tree-growing set, 60 images made up the holdout set, and the remaining 63 images were used for evaluation. Table 3 summarizes the results, using the same notation as described in Table 1. While the true positive and false positive rates do vary across settings of the parameter, in each case the filter retains a high percentage of edge pixels on the object (roughly 71 to 83 percent) while removing roughly 90 percent of all background edge pixels. Exactly how the setting of \( \sigma \) affects true positive and false positive rates is complex and depends on characteristics of the viewing environment, the objects present, the classifiers in the cascade, and the policy for arranging the relative probe centers in the aperture. Nonetheless, these results suggest that the classifier cascade performs well over a range of reasonable values for this parameter.

### 4.2.4 Complexity

This section addresses concerns about the time and space complexity of our edge filtering approach, in terms of both the training and test phases. In particular, we demonstrate that although a potentially large number of edge features is considered by our algorithm and, although we employ numerous classifiers at training and test time, the amount of computation required to train the cascade and evaluate a novel image is feasible.

Training the cascade of classifiers involves computing edge probes and inducing a decision tree for each phase of the cascade. At a particular cascade phase, an exhaustive set of edge probes is computed over all edge pixels in the training set; thus, space requirements at training time will be determined by the number of edge pixels in the training set at each cascade phase, along with the number of relative probe centers in the aperture at each phase. In Table 4, we show these numbers for a few phases of a classifier cascade trained on images of the ladder in the classroom environment (Fig. 8, third row) as described in Section 4.2.1. This table also shows approximate running times of the decision tree inducer on a 1.67 GHz Athlon for those cascade phases and the number of nodes in the learned decision trees. On one hand, due to our strategy of evenly spacing the relative probe centers in the aperture, the dimensionality of the training data increases as the aperture is grown; on the other hand, since we filter training examples out of the training set at each phase of cascade training, the number of training examples decreases as the aperture is grown. The total time required to train one classifier cascade on a 1.67 GHz machine, including all decision tree induction and edge probe calculation, is approximately one day.

When evaluating a novel image, some number of edge probes are computed at each edge pixel in the image until either the edge pixel is filtered out of the image or the last phase of the cascade is reached. Thus, the time complexity of evaluating a novel image will in large part be determined by the number of edge probes required for each of its edge pixels. To get a sense of the total number of edge probes computed at typical edge pixels in test images, we took the classifier cascade trained on images of the ladder in the living room (Fig. 8, sixth row) as described in Section 4.2.1, and counted the number of edge probes required to classify each edge pixel in each test image of the living room, over all cascade phases. Fig. 9 summarizes these edge probe counts in a histogram and cumulative distribution. Note that roughly 23 percent of all edge pixels are classified based on ten or fewer distinct edge probes, and 95 percent of all edge pixels in the test images require evaluation of 50 or fewer edge probes. This is significant since the total number of relative probe centers in the largest aperture (Table 4, last row) is 1,309. Thus, while the training phase selects edge features from a large set of potential features, relatively few of these features are evaluated for any given image pixel at run time (see Fig. 10). Given detected edges, we found that it took about one second on a standard desktop PC to classify all the edge pixels in the 1,600-by-1,200 ladder images using our implementation.
In this section, we explore what features our decision trees compute in order to classify edge pixels. Since decision trees classify training examples by making a series of threshold tests on individual features, we can traverse the tree in order to interpret it in terms of a step-by-step analysis of what tests it is making in order to arrive at a classification. We stress that our decision trees were not designed at the outset with interpretability in mind; for tree-building techniques along these lines, see for example [12]. However, by manually tracing paths through our trees, we can in some cases gain insight into the visual characteristics being exploited for discrimination. To do so, we examined decision trees induced on the chair data set (Section 4.1) and searched for leaves containing large numbers of training examples. For each of those leaves, we traced the path through the tree leading from the root node to the leaf and made a graphical representation of the features being tested at each node and the values of the split.

More specifically, consider representing each node in the decision tree as a pair \((\delta, t)\); a pixel \(q\) is sent down the left branch at that node if \(ep(q + \delta, G) > t\) and down the right branch of the tree if \(ep(q + \delta, G) < t\). A path from root to leaf generates a set of these pairs, each pair corresponding to a test at a node along the path. These tests define a set of intervals in which the edge probe values must lie in order for the test pixel to arrive at the leaf. That is, for each relative probe center \(\delta\), we can define a set of intervals \([t_{\text{min}}, t_{\text{max}}]\) such that, in order to be classified into the leaf in question, \(ep(q + \delta, G)\) must be greater than \(t_{\text{min}}\) and less than \(t_{\text{max}}\). We represent these valid feature value intervals graphically by plotting the range between the maximum and minimum edge probe values over all edge probes, and the subranges \([t_{\text{min}}, t_{\text{max}}]\) for each \(\delta\). In Fig. 11, the locations of relative probe centers \(\delta\) is plotted in red, the query edge pixel \(q\) is in yellow, the range of all edge probe values is in grey, and the subrange \([t_{\text{min}}, t_{\text{max}}]\) is in black. High edge probe values correspond to higher points in the gray range (i.e., points closer to the top of the page). Due to scaling issues, only the bottom-most tip of the range corresponds to a low concentration of edge pixels in the image.

Example displays of admissible feature value intervals for two different decision tree paths are shown in Fig. 11. In Fig. 11a, we show feature intervals corresponding to a highly populated leaf leading to a positive classification. The display suggests that edge concentration should be high for probe centers lying on a diagonal line leading from below and left of the pixel, to above and right of the pixel. Also, edge probe values should be low for probe centers on either side of that line. This suggests that this particular leaf characterizes edge pixels lying along a diagonal edge that is more or less isolated from the background environment. Indeed, in several of the training images, the legs of the chair are in this configuration, and we show an example in Fig. 11b. Fig. 11c shows admissible ranges of values for edge probes leading to a negative classification. Edge probes evaluated at three probe centers at close proximity above, below, and to one side of the pixel must be in the high range of values in order for the pixel to arrive in the leaf. This suggests that edge pixels with very high edge density all around them are likely to belong to the background. In fact, most pixels in regions so busy with edges do in fact project to the background (see Fig. 11d for an example).

6 CONCLUSION
Our approach to separating objects from their background environment based on edge cues consists of screening each edge pixel in the image through a series of classifiers, each of which computes sets of edge features over successively larger image areas. Each classifier in the cascade computes a sparse set of localized edge features in a sequence determined by its tree structure. By tuning feature extraction to the object and background present in training
images, we overcome the effects of object structure (concavities, holes, wiry structures) that tend to confound template-based approaches to recognition. And by screening the image through a series of increasingly complex classifiers, we quickly discard edge pixels that are easily discriminated from the object, saving computation for more ambiguous portions of the image.

The algorithm presented in this paper takes a binary edge image as input and derives spatial features of those edges as a cue for recognition; an underlying assumption is that the edge detector has been properly tuned to extract edges from the input image. An alternative that we will explore in future work is to take a raw image as input and derive both the edge detector and spatial features at training time; this will obviate the need to hand-tune an edge detector before applying our algorithm. Furthermore, we formulate our problem in terms of classifying each edge pixel independently, making no attempt to ensure that neighboring object edge pixels correspond to neighboring locations on the object itself. Another avenue for future research is to use training data to enforce this sort of spatial coherence across edge pixels identified as belonging to the target object. For example, the Markov random field techniques employed in [30], [8] can enforce this spatial coherence.

ACKNOWLEDGMENTS
This research was supported in part by US National Science Foundation Grant IIS-9907142. Earlier versions of this work appeared in [13] and [14].

REFERENCES
[1] H. Almuallim and T.G. Dietterich, “Learning with Many Irrelevant Features,” Proc. Nat’l Conf. Artificial Intelligence (AAAI), 1991.
[2] Y. Amit, D. Geman, and K. Wilder, Joint Induction of Shape Features and Tree Classifiers, 1997.

[3] R. Basri and D. Jacobs, "Projective Alignment with Regions," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 5, pp. 519-527, May 2001.

[4] J. Beis and D. Lowe, "Indexing without Invariants in 3D Object Recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 21, no. 10, pp. 1000-1015, Oct. 1999.

[5] P. Belhumeur and D. Kriegman, "What is the Set of Images of an Object under All Possible Illumination Conditions?" Int'l J. Comp. Vision, vol. 28, no. 3, pp. 245-260, 1998.

[6] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 509-522, Apr. 2002.

[7] J. Ben-Arie, Z. Wang, and R. Rao, "Iconic Recognition with Affine-Invariant Spectral Signatures," Proc. IAPR-IEEE Int'l Conf. Pattern Recognition, vol. 1, pp. 672-676, 1996.

[8] Y. Boykov and D.P. Huttenlocher, "A New Bayesian Framework for Object Recognition," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 517-523, 1999.

[9] J.P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C.E. Brodley, "Pruning Decision Trees with Misclassification Costs," Proc. European Conf. Machine Learning, pp. 131-136, Apr. 1998.

[10] A.P. Bradley, "The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms," Pattern Recognition, vol. 30, no. 7, pp. 1145-1159, 1997.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees. Belmont, Calif.: Wadsworth Int'l, 1984.

[12] A. Buja and Y.-S. Lee, "Data Mining Criteria for Tree-Based Regression and Classification," Proc. Int'l Conf. Knowledge Discovery and Data Mining, 2001.

[13] O. Carmichael and M. Hebert, "Object Recognition by a Cascade of Edge Probes," Proc. British Machine Vision Conf., 2002.

[14] O. Carmichael and M. Hebert, "Shape-Based Recognition of Wiry Objects," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2003.

[15] O. Carmichael and M. Hebert, "Word: Wiry Object Recognition Database," Carnegie Mellon Univ., www.cs.cmu.edu/owen/word, Jan. 2004.

[16] V. Colin de Verdiere and J.L. Crowley, "Visual Recognition Using Local Appearance," Proc. European Conf. Computer Vision, pp. 640-654, 1998.

[17] R. Duda, P. Hart, and D. Stork, Pattern Classification, second ed. Wiley-Interscience, 2001.

[18] J.P. Egan, Signal Detection Theory and ROC Analysis. Series in Cognition and Perception, New York: Academic Press, 1975.

[19] U.M. Fayyad and K.B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," Proc. Int'l Joint Conf. Artificial Intelligence, pp. 800-805, 1989.

[20] R. Fergus, P. Perona, and A. Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning," Proc. IEEE Conf. Computer Vision And Pattern Recognition, vol. 2, pp. 264-271, 2003.

[21] W.T. Freeman, "Exploiting the Generic Viewpoint Assumption," Int'l J. Computer Vision, vol. 20, no. 3, pp. 243-261 1996.

[22] W.E.L. Grimson, Object Recognition by Computer: The Role of Geometric Constraints. MIT Press, 1990.

[23] W.E.L. Grimson and D. Huttenlocher, "On the Sensitivity of Geometric Hashing," Proc. Int'l Conf. Computer Vision, pp. 334-338, 1990.

[24] D. Keren, M. Osadchy, and C. Gotsman, "Antifaces: A Novel, Fast Method for Image Detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 7, pp. 747-781, July 2001.

[25] K. Kira and L.A. Rendell, "A Practical Approach to Feature Selection," Proc. Int'l Conf. Machine Learning, pp. 249-256 1992.

[26] R. Kohavi, D. Sommerfield, and J. Dougherty, "Data Mining Using MLC++; A Machine Learning Library in C++," Tools with Artificial Intelligence, 1996, http://www.sgi.com/tech/mlc.

[27] I. Kononenko and E. Simec, Induction of Decision Trees Using RELIEF. 1995.

[28] J. Krumm, "Object Detection with Vector Quantized Binary Features," Proc. IEEE Conf. Computer Vision And Pattern Recognition, pp. 179-185, June 1997.

[29] Y. Lambdan and H. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," Proc. Int'l Conf. Computer Vision, pp. 238-249, 1988.

[30] S.Z. Li and J. Hornegger, "A Two-Stage Probabilistic Approach for Object Recognition," Proc. European Conf. Computer Vision, 1998.

[31] S. Li, L. Zhu, Z.Q. Zhang, A. Blake, H.J. Zhang, and H. Shum, "Statistical Learning of Multi-View Face Detection," Proc. European Conf. Computer Vision, 2002.

[32] R. Lienhart, L. Liang, and A. Kuranov, "A Detector Tree of Boosted Classifiers for Real-Time Object Detection and Tracking," IEEE Int'l Conf. Multimedia and Systems (ICME2003), July 2003.

[33] D. Lowe, "Three-Dimensional Object Recognition from Single Two-Dimensional Images," Artificial Intelligence, vol. 31, pp. 355-395, 1987.

[34] D. Lowe, "Object Recognition from Local Scale-Invariant Features," Proc. Int'l Conf. Computer Vision, pp. 1150-1157 1999.

[35] D.G. Lowe, Perceptual Organization and Visual Recognition. Kluwer Academic, 1985.

[36] K. Mikolajczyk, A. Zisserman, and C. Schmid, "Shape Recognition with Edge-Based Features," Proc. British Machine Vision Conf., 2003.

[37] T. Mitchell, Machine Learning, first ed. WCB/McGraw-Hill 1997.

[38] H. Murase and S. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance," Int'l J. Computer Vision, vol. 14, pp. 8-24, 1995.

[39] A. Pope and D. Lowe, "Vista: A Software Environment for Computer Vision Research," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1994.

[40] F.J. Provost, T. Fawcett, and R. Kohavi, "The Case Against Accuracy Estimation for Comparing Induction Algorithms," Proc. Int'l Conf. Machine Learning, pp. 445-453, 1998.

[41] J.R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[42] H. Schneiderman and T. Kanade, "Object Detection Using the Statistics of Parts," Int'l J. Computer Vision, 2002.

[43] A. Selinger and R.C. Nelson, "A Perceptual Grouping Hierarchy for Appearance-Based 3D Object Recognition," Computer Vision and Image Understanding, vol. 76, no. 1, pp. 83-92 1999.

[44] A. Thiruvaanaanathan, B. Stenger, P.H.S. Torr, and R. Cipolla, "Shape Context and Chamfer Matching in Cluttered Scenes," Proc. IEEE Conf. Computer Vision And Pattern Recognition, 2003.

[45] P. Viola and W.M. Wells III, "Alignment by Maximization of Mutual Information," Int'l J. Computer Vision, vol. 24, no. 2, pp. 137-154 1997.

[46] P. Viola and M. Jones, "Robust Real-Time Object Detection," Technical Report CRL. 2001/01, Compaq Cambridge Research Laboratory, 2001.

[47] J. Wu, J.M. Rehg, and M.D. Mullin, "Learning a Rare Event Detection Cascade by Direct Feature Selection," Advances in Neural Information Processing Systems (NIPS), 2003.

[48] B. Zadrozny and C. Elkan, "Learning and Making Decisions When Costs and Probabilities are Both Unknown," Knowledge Discovery and Data Mining, pp. 204-213, 2001.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.