Structured Forests for Fast Edge Detection

Piotr Dollár
Microsoft Research
pdollar@microsoft.com

C. Lawrence Zitnick
Microsoft Research
larryz@microsoft.com

Abstract

Edge detection is a critical component of many vision systems, including object detectors and image segmentation algorithms. Patches of edges exhibit well-known forms of local structure, such as straight lines or T-junctions. In this paper we take advantage of the structure present in local image patches to learn both an accurate and computationally efficient edge detector. We formulate the problem of predicting local edge masks in a structured learning framework applied to random decision forests. Our novel approach to learning decision trees robustly maps the structured labels to a discrete space on which standard information gain measures may be evaluated. The result is an approach that obtains realtime performance that is orders of magnitude faster than many competing state-of-the-art approaches, while also achieving state-of-the-art edge detection results on the BSDS500 Segmentation dataset and NYU Depth dataset. Finally, we show the potential of our approach as a general purpose edge detector by showing our learned edge models generalize well across datasets.

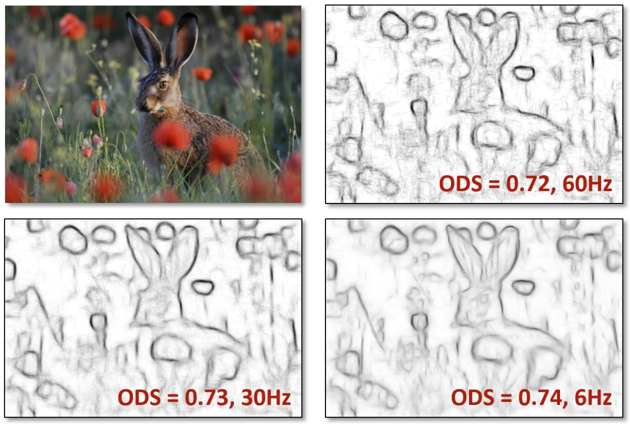

Figure 1. Edge detection results using three versions of our Structured Edge (SE) detector demonstrating tradeoffs in accuracy vs. runtime. We obtain realtime performance while simultaneously achieving state-of-the-art results. ODS numbers were computed on BSDS [1] on which the highly tuned gPb detector [1] achieves a score of .73. The variants shown include SE-SS ($T = 1$), SE-SS ($T = 4$), and SE-MS, see §4 for details.

1. Introduction

Edge detection has remained a fundamental task in computer vision since the early 1970's [13, 10, 32]. The detection of edges is a critical preprocessing step for a variety of tasks, including object recognition [36, 12], segmentation [23, 1], and active contours [19]. Traditional approaches to edge detection use a variety of methods for computing color gradient magnitudes followed by non-maximal suppression [5, 14, 38]. Unfortunately, many visually salient edges do not correspond to color gradients, such as texture edges [24] and illusory contours [29]. State-of-the-art approaches to edge detection [1, 31, 21] use a variety of features as input, including brightness, color and texture gradients computed over multiple scales. For top accuracy, globalization based on spectral clustering may also be performed [1, 31].

Since visually salient edges correspond to a variety of visual phenomena, finding a unified approach to edge detection is difficult. Motivated by this observation several recent papers have explored the use of learning techniques for edge detection [9, 37, 21]. Each of these approaches takes an image patch and computes the likelihood that the center pixel contains an edge. The independent edge predictions may then be combined using global reasoning [37, 1, 31].

The edges in a local patch are highly interdependent [21]. They often contain well-known patterns, such as straight lines, parallel lines, T-junctions or Y-junctions [30, 21]. Recently, a family of learning approaches called *structured learning* [26] has been applied to problems exhibiting similar characteristics. For instance, [20] applies structured learning to the problem of semantic image labeling for which local image labels are also highly interdependent.

In this paper we propose a generalized structured learning approach that we apply to edge detection. This approach allows us to take advantage of the inherent structure in edge patches, while being surprisingly computationally efficient. We can compute edge maps in realtime, which is orders of magnitude faster than competing state-of-the-art approaches. A random forest framework is used to capture the structured information [20]. We formulate the problem of edge detection as predicting local segmentation masks given input image patches. Our novel approach to learning decision trees uses structured labels to determine the splitting function at each branch in the tree. The structured labels are robustly mapped to a discrete space on which standard information gain measures may be evaluated. Each forest predicts a patch of edge pixel labels that are aggregated across the image to compute our final edge map, see Figure 1. We show state-of-the-art results on both the BSDS500 [1] and the NYU Depth dataset [33, 31]. We demonstrate the potential of our approach as a general purpose edge detector by showing the strong cross dataset generalization of our learned edge models.

1.1. Related work

In this section we discuss related work in edge detection and structured learning.

Edge detection: Numerous papers have been written on edge detection over the past 50 years. Early work [13, 10, 5, 27, 14] focused on the detection of intensity or color gradients. The popular Canny detector [5] finds the peak gradient magnitude orthogonal to the edge direction. An evaluation of various low-level edge detectors can be found in [3] and an overview in [38]. More recently, the works of [24, 22, 1] explore edge detection in the presence of textures.

Several techniques have explored the use of learning for edge detection [9, 37, 22, 31, 21]. Dollár et al. [9] used a boosted classifier to independently label each pixel using its surrounding image patch as input. Zheng et al. [37] combine low-, mid- and high-level cues and show improved results for object-specific edge detection. Recently, Ren and Bo [31] improved the result of [1] by computing gradients across learned sparse codes of patch gradients. While [31] achieved state-of-the-art results, their approach further increased the high computational cost of [1]. Catanzaro et al. [6] improve the runtime of [1] using parallel algorithms.

Finally, Lim et al. [21] propose an edge detection approach that classifies edge patches into *sketch tokens* using random forest classifiers, that, like in our work, attempt to capture local edge structure. Sketch tokens bear resemblance to earlier work on *shapemes* [30] but are computed directly from color image patches rather than from pre-computed edge maps. The result is an efficient approach for detecting edges that also shows promising results for object detection. In contrast to previous work, we do not require the use of pre-defined classes of edge patches. This allows us to learn more subtle variations in edge structure and leads to a more accurate and efficient algorithm.

Structured learning: Structured learning addresses the problem of learning a mapping where the input or output space may be arbitrarily complex representing strings, sequences, graphs, object pose, bounding boxes etc. [35, 34, 2]. We refer readers to [26] for a comprehensive survey.

Our structured random forests differ from these works in several respects. First, we assume that only the output space is structured and operate on a standard input space. Second, by default our model can only output examples observed during training (although this can be ameliorated with custom ensemble models). On the other hand, common approaches for structured prediction learn parameters to a scoring function, and to obtain a prediction, an optimization over the output space must be performed [35, 26]. This requires defining a scoring function and an efficient (possibly approximate) optimization procedure. In contrast, inference using our structured random forest is straightforward, general and fast (same as for standard random forests).

Finally, our work was inspired by the recent paper by Kontschieder et al. [20] on learning random forests for structured class labels for the specific case where the output labels represent a semantic image labeling for an image patch. The key observation made by Kontschieder et al. is that given a color image patch, the leaf node reached in a tree is independent of the structured semantic labels, and any type of output can be stored at each leaf. Building on this idea, we propose a general learning framework for structured output forests that can be used with a broad class of output spaces and we apply our framework to learning an accurate and fast edge detector.

2. Random Decision Forests

We begin with a review of random decision forests [4, 15]. Throughout our presentation we adopt the notation and terminology of the extensive recent survey by Criminisi et al. [7], somewhat simplified for ease of presentation. The notation in [7] is sufficiently general to support our extension to random forests with structured outputs.

A decision tree $f_t(x)$ classifies a sample $x \in \mathcal{X}$ by recursively branching left or right down the tree until a leaf node is reached. Specifically, each node j in the tree is associated with a binary *split function*:

$$h(x, \theta_j) \in \{0, 1\} \tag{1}$$

with parameters θ_j. If $h(x, \theta_j) = 0$ node j sends x left, otherwise right, with the process terminating at a leaf node. The output of the tree on an input x is the prediction stored at the leaf reached by x, which may be a target label $y \in \mathcal{Y}$ or a distribution over the labels $\mathcal{Y}$.

While the split function $h(x, \theta)$ may be arbitrarily complex, a common choice is a 'stump' where a single feature dimension of x is compared to a threshold. Specifically, $\theta = (k, \tau)$ and $h(x, \theta) = [x(k) < \tau]$, where $[\cdot]$ denotes the indicator function. Another popular choice is $\theta = (k_1, k_2, \tau)$ and $h(x, \theta) = [x(k_1) - x(k_2) < \tau]$. Both are computationally efficient and effective in practice [7].

A decision forest is an ensemble of T independent trees f_t. Given a sample x, the predictions $f_t(x)$ from the set of trees are combined using an *ensemble model* into a single output. Choice of ensemble model is problem specific and depends on $\mathcal{Y}$, common choices include majority voting for classification and averaging for regression, although more sophisticated ensemble models may be employed [7].

Observe that arbitrary information may be stored at the leaves of a decision tree. The leaf node reached by the tree depends only on the input x, and while predictions of multiple trees must be merged in some useful way (the ensemble model), any type of output y can be stored at each leaf. This allows use of complex output spaces $\mathcal{Y}$, including structured outputs as observed by Kontschieder et al. [20].

While prediction is straightforward, training random decision forests with structured $\mathcal{Y}$ is more challenging. We review the standard learning procedure next and describe our generalization to learning with structured outputs in §3.

2.1. Training Decision Trees

Each tree is trained independently in a recursive manner. For a given node j and training set $\mathcal{S}_j \subset \mathcal{X} \times \mathcal{Y}$, the goal is to find parameters θ_j of the split function $h(x, \theta_j)$ that result in a 'good' split of the data. This requires defining an *information gain criterion* of the form:

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R) \qquad (2)$$

where $\mathcal{S}_j^L = \{(x, y) \in \mathcal{S}_j | h(x, \theta_j) = 0\}$, $\mathcal{S}_j^R = \mathcal{S}_j \backslash \mathcal{S}_j^L$. Splitting parameters θ_j are chosen to maximize the information gain I_j; training then proceeds recursively on the left node with data $\mathcal{S}_j^L$ and similarly for the right node. Training stops when a maximum depth is reached or if information gain or training set size fall below fixed thresholds.

For multiclass classification ($\mathcal{Y} \subset \mathbb{Z}$) the standard definition of information gain can be used:

$$I_j = H(\mathcal{S}_j) - \sum_{k \in \{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k) \qquad (3)$$

where $H(\mathcal{S}) = -\sum_y p_y \log(p_y)$ denotes the Shannon entropy and p_y is the fraction of elements in $\mathcal{S}$ with label y. Alternatively the Gini impurity $H(\mathcal{S}) = \sum_y p_y(1-p_y)$ has also been used in conjunction with Eqn. (3) [4].

For regression, entropy and information gain can be extended to continuous variables [7]. Alternatively, a common approach for single-variate regression ($\mathcal{Y} = \mathbb{R}$) is to minimize the variance of labels at the leaves [4]. If we write the variance as $H(S) = \frac{1}{|\mathcal{S}|} \sum_y (y - \mu)^2$ where $\mu = \frac{1}{|\mathcal{S}|} \sum_y y$, then substituting H for entropy in Eqn. (3) leads to the standard criterion for single-variate regression.

Can we define a more general *information gain* criterion for Eqn. (2) that applies to arbitrary output spaces $\mathcal{Y}$? Surprisingly yes, given mild additional assumptions about $\mathcal{Y}$. Before going into detail in §3, we discuss the key role that randomness plays in training of decision forests next.

2.2. Randomness and Optimality

Individual decision trees exhibit high variance and tend to overfit [17, 4, 15]. Decision forests ameliorate this by training multiple de-correlated trees and combining their output. A crucial component of the training procedure is therefore to achieve a sufficient diversity of trees.

Diversity of trees can be obtained either by randomly subsampling the data used to train each *tree* [4] or randomly subsampling the features and splits used to train each *node* [17]. Injecting randomness at the level of nodes tends to produce higher accuracy models [15] and has proven more popular [7]. Specifically, when optimizing Eqn. (2), only a small set of possible θ_j are sampled and tested when choosing the optimal split. E.g., for stumps where $\theta = (k, \tau)$ and $h(x, \theta) = [x(k) < \tau]$, [15] advocates sampling $\sqrt{d}$ features where $\mathcal{X} = \mathbb{R}^d$ and a single threshold τ per feature.

In effect, accuracy of individual trees is sacrificed in favor of a high diversity ensemble [15]. Leveraging similar intuition allows us to introduce an approximate information gain criterion for structured labels, described next, and leads to our generalized structured forest formulation.

3. Structured Random Forests

In this section we extend random decision forests to general structured output spaces $\mathcal{Y}$. Of particular interest for computer vision is the case where $x \in \mathcal{X}$ represents an image patch and $y \in \mathcal{Y}$ encodes the corresponding local image annotation (e.g., a segmentation mask or set of semantic image labels). However, we keep our derivation general.

Training random forests with structured labels poses two main challenges. First, structured output spaces are often high dimensional and complex. Thus scoring numerous candidate splits directly over structured labels may be prohibitively expensive. Second, and more critically, information gain over structured labels may not be well defined.

We use the observation that even *approximate* measures of information gain suffice to train effective random forest classifiers [15, 20]. 'Optimal' splits are not necessary or even desired, see §2.2. Our core idea is to map all the structured labels $y \in \mathcal{Y}$ at a given node into a discrete set of labels $c \in \mathcal{C}$, where $\mathcal{C} = \{1, \ldots, k\}$, such that *similar* structured labels y are assigned to the same discrete label c.

Given the discrete labels $\mathcal{C}$, information gain calculated directly and efficiently over $\mathcal{C}$ can serve as a proxy for the information gain over the structured labels $\mathcal{Y}$. By mapping the structured labels to discrete labels prior to training each node we can leverage existing random forest training procedures to learn structured random forests effectively.

Our approach to calculating information gain relies on measuring similarity over $\mathcal{Y}$. However, for many structured output spaces, including those used for edge detection, computing similarity over $\mathcal{Y}$ is not well defined. Instead, we define a mapping of $\mathcal{Y}$ to an *intermediate* space $\mathcal{Z}$ in which distance is easily measured. We therefore utilize a broadly applicable two-stage approach of first mapping $\mathcal{Y} \rightarrow \mathcal{Z}$ followed by a straightforward mapping of $\mathcal{Z} \rightarrow \mathcal{C}$.

We describe the proposed approach in more detail next and return to its application to edge detection in §4.

3.1. Intermediate Mapping Π

Our key assumption is that for many structured output spaces, including for structured learning of edge detection, we can define a mapping of the form:

$$\Pi : \mathcal{Y} \rightarrow \mathcal{Z} \tag{4}$$

such that we can approximate dissimilarity of $y \in \mathcal{Y}$ by computing Euclidean distance in $\mathcal{Z}$. For example, as we describe in detail in §4, for edge detection the labels $y \in \mathcal{Y}$ are 16×16 segmentation masks and we define $z = \Pi(y)$ to be a long binary vector that encodes whether every pair of pixels in y belong to the same or different segments. Distance is easily measured in the resulting space $\mathcal{Z}$.

$\mathcal{Z}$ may be high dimensional which presents a challenge computationally. For example, for edge detection there are $\binom{16\cdot16}{2} = 32640$ unique pixel pairs in a 16×16 segmentation mask, so computing z for every y would be expensive. However, as only an approximate distance measure is necessary, the dimensionality of $\mathcal{Z}$ can be reduced.

In order to reduce dimensionality, we sample m dimensions of $\mathcal{Z}$, resulting in a reduced mapping $\Pi_\phi : \mathcal{Y} \rightarrow \mathcal{Z}$ parametrized by ϕ. During training, a distinct mapping Π_ϕ is randomly generated and applied to training labels $\mathcal{Y}_j$ at each node j. This serves two purposes. First, Π_ϕ can be considerably faster to compute than Π. Second, sampling $\mathcal{Z}$ injects additional randomness into the learning process and helps ensure a sufficient diversity of trees, see §2.2.

Finally, Principal Component Analysis (PCA) [18] can be used to further reduce the dimensionality of $\mathcal{Z}$. PCA denoises $\mathcal{Z}$ while approximately preserving Euclidean distance. In practice, we use Π_ϕ with $m = 256$ dimensions followed by a PCA projection to at most 5 dimensions.

3.2. Information Gain Criterion

Given the mapping $\Pi_\phi : \mathcal{Y} \rightarrow \mathcal{Z}$, a number of choices for the information gain criterion are possible. For discrete $\mathcal{Z}$ multi-variate joint entropy could be computed directly. Kontschieder et al. [20] proposed such an approach, but due to its complexity of $O(|\mathcal{Z}|^m)$, were limited to using $m \leq 2$. Our experiments indicate $m \geq 64$ is necessary to accurately capture similarities between elements in $\mathcal{Z}$. Alternatively, given continuous $\mathcal{Z}$, variance or a continuous formulation of entropy [7] can be used to define information gain. In this work we propose a simpler, extremely efficient approach.

We map a set of structured labels $y \in \mathcal{Y}$ into a discrete set of labels $c \in \mathcal{C}$, where $\mathcal{C} = \{1, \ldots, k\}$, such that labels with similar z are assigned to the same discrete label c. The discrete labels may be binary ($k = 2$) or multi-class ($k > 2$). This allows us to use standard information gain criteria based on Shannon entropy or Gini impurity as defined in Eqn. (3). Critically, discretization is performed independently when training each node and depends on the distribution of labels at a given node (contrast with [21]).

We consider two straightforward approaches to obtaining the discrete label set $\mathcal{C}$ given $\mathcal{Z}$. Our first approach is to cluster z into k clusters using K-means. Alternatively, we quantize z based on the top $\log_2(k)$ PCA dimensions, assigning z a discrete label c according to the orthant (generalization of quadrant) into which z falls. Both approaches perform similarly but the latter is slightly faster. We use PCA quantization with $k = 2$ in all experiments.

3.3. Ensemble Model

Finally, we need to define how to combine a set of n labels $y_1 \ldots y_n \in \mathcal{Y}$ into a single prediction both for training (to associate labels with nodes) and testing (to merge multiple predictions). As before, we sample an m dimensional mapping Π_ϕ and compute $z_i = \Pi_\phi(y_i)$ for each i. We select the label y_k whose z_k is the medoid, i.e. the z_k that minimizes the sum of distances to all other z_i[1].

The ensemble model depends on m and the selected mapping Π_ϕ. However, we only need to compute the medoid for small n (either for training a leaf node or merging the output of multiple trees), so having a coarse distance metric suffices to select a representative element y_k.

The biggest limitation is that any prediction $y \in \mathcal{Y}$ must have been observed during training; the ensemble model is unable to synthesize novel labels. Indeed, this is impossible without additional information about $\mathcal{Y}$. In practice, domain specific ensemble models can be preferable. For example, in edge detection we use the default ensemble model during training but utilize a custom approach for merging outputs over multiple overlapping image patches.

4. Edge Detection

In this section we describe how we apply our structured forest formulation to the task of edge detection. As input our method takes an image that may contain multiple channels, such as an RGB or RGBD image. The task is to label each pixel with a binary variable indicating whether the pixel contains an edge or not. Similar to the task of semantic image labeling [20], the labels within a small image patch are highly interdependent, providing a promising candidate problem for our structured forest approach.

We assume we are given a set of segmented training images, in which the boundaries between the segments correspond to contours [1, 33]. Given an image patch, its annotation can be specified either as a *segmentation mask* indicating segment membership for each pixel (defined up to a permutation) or a binary *edge map*. We use $y \in \mathcal{Y} = \mathbb{Z}^{d \times d}$ to denote the former and $y' \in \mathcal{Y}' = \{0, 1\}^{d \times d}$ for the latter, where d indicates patch width. An edge map y' can always be trivially derived from segmentation mask y, but not vice versa. We utilize both representations in our approach.

[1]The medoid z_k minimizes $\sum_{ij}(z_{kj} - z_{ij})^2$. This is equivalent to $\min_k \sum_j (z_{kj} - \bar{z}_j)^2$ and can be computed efficiently in time $O(nm)$.

Next, we describe how we compute the input features x, the mapping functions Π_ϕ used to determine splits, and the ensemble model used to combine multiple predictions.

Input features: Our learning approach predicts a structured 16×16 segmentation mask from a larger 32×32 image patch. We begin by augmenting each image patch with multiple additional *channels* of information, resulting in a feature vector $x \in \mathbb{R}^{32\times32\times K}$ where K is the number of channels. We use features of two types: pixel lookups $x(i,j,k)$ and pairwise differences $x(i_1,j_1,k) - x(i_2,j_2,k)$, see §2.

Inspired by the edge detection results of Lim et al. [21], we use a similar set of color and gradient channels (originally developed for fast pedestrian detection [8]). We compute 3 color channels in CIE-LUV color space along with normalized gradient magnitude at 2 scales (original and half resolution). Additionally, we split each gradient magnitude channel into 4 channels based on orientation. The channels are blurred with a triangle filter of radius 2 and downsampled by a factor of 2. The result is 3 color, 2 magnitude and 8 orientation channels, for a total of 13 channels.

We downsample the channels by a factor of 2, resulting in $32 \cdot 32 \cdot 13/4 = 3328$ candidate features $x(i,j,k)$. Motivated by [21], we also compute pairwise difference features. We apply a large triangle blur to each channel (8 pixel radius), and downsample to a resolution of 5×5. Sampling all candidate pairs and computing their differences yields an additional $\binom{5\cdot5}{2} = 300$ candidate features per channel, resulting in 7228 total candidate features per patch.

Mapping function: To train decision trees, we need to define a mapping $\Pi : \mathcal{Y} \to \mathcal{Z}$ as described in §3. Recall that our structured labels y are 16×16 segmentation masks. One option is to use $\Pi : \mathcal{Y} \to \mathcal{Y}'$, where y' represents the binary edge map corresponding to y. Unfortunately Euclidean distance over $\mathcal{Y}'$ yields a brittle distance measure.

We therefore define an alternate mapping Π. Let $y(j)$ for $1 \le j \le 256$ denote the j^{th} pixel of mask y. Since y is defined only up to a permutation, a single value $y(j)$ yields no information about y. Instead we can sample a pair of locations $j_1 \neq j_2$ and check if $y(j_1) = y(j_2)$. This allows us to define $z = \Pi(y)$ as a large binary vector that encodes $[y(j_1) = y(j_2)]$ for every unique pair of indices $j_1 \neq j_2$. While $\mathcal{Z}$ has $\binom{256}{2}$ dimensions, in practice we only compute a subset of m dimensions as discussed in §3.2. We found a setting of $m = 256$ and $k = 2$ gives good results, effectively capturing the similarity of segmentation masks.

Ensemble model: Random forests achieve robust results by combining the output of multiple decorrelated trees. While merging multiple segmentation masks $y \in \mathcal{Y}$ is difficult, multiple edge maps $y' \in \mathcal{Y}'$ can be averaged to yield a soft edge response. Taking advantage of a decision tree's ability to store arbitrary information at the leaf nodes, in addition to the learned segmentation mask y we also store the corresponding edge map y'. This allows the predictions of multiple trees to be combined through averaging.

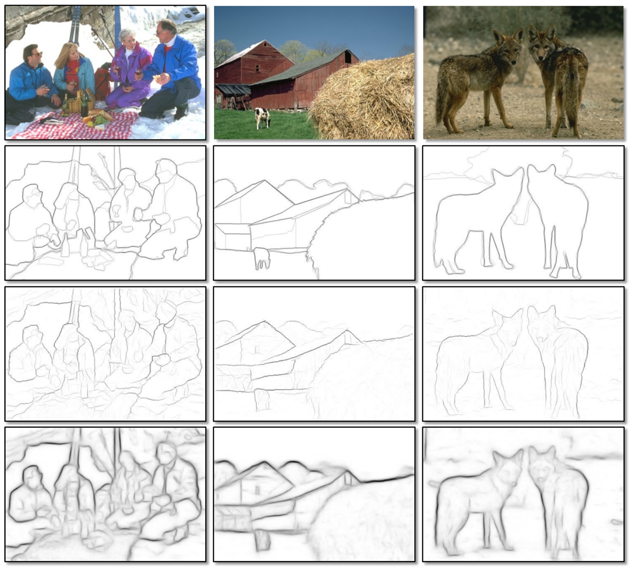
Figure 2. Illustration of edge detection results on the BSDS500 dataset: (top row) original image, (second row) ground truth, (third row) results of SCG [31], and (last row) our results for SE-MS.

The surprising efficiency of our approach derives from the use of structured labels that capture information for an entire image neighborhood, reducing the number of decision trees T that need to be evaluated per pixel. We compute our structured output densely on the image with a stride of 2 pixels, thus with 16×16 output patches, each pixel receives $16^2T/4 \approx 64T$ predictions. In practice we use $1 \le T \le 4$.

A critical assumption is that predictions are uncorrelated. Since both the inputs and outputs of each tree overlap, we train $2T$ total trees and evaluate an alternating set of T trees at each adjacent location. Use of such a 'checkerboard pattern' improves results somewhat, introducing larger separation between the trees did not improve results further.

Multiscale detection: Inspired by the work of Ren [28], we implement a multiscale version of our edge detector. Given an input image I, we run our structured edge detector on the original, half, and double resolution version of I and average the result of the three edge maps after resizing to the original image dimensions. Although somewhat inefficient, the approach noticeably improves edge quality.

Parameters: All parameters were set using validation sets fully independent of the test sets; *detailed experiments are reported in the supplementary material.* Parameters include: image and label patch size, channel and feature parameters (e.g., image and channel blurring), and decision forest parameters (stopping criteria, number of trees, m and k). Each tree was trained with one million randomly sampled patches. Training takes ~30 minutes per tree and is parallelized over trees. Evaluation of trees is parallelized as well, we use a quad-core machine for all reported runtimes.

	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.60	.64	.58	15
Felz-Hutt [11]	.61	.64	.56	10
Hidayat-Green [16]	.62[†]	-	-	20
BEL [9]	.66[†]	-	-	1/10
gPb + GPU [6]	.70[†]	-	-	1/2[‡]
gPb [1]	.71	.74	.65	1/240
gPb-owt-ucm [1]	.73	**.76**	.73	1/240
Sketch tokens [21]	.73	.75	**.78**	1
SCG [31]	**.74**	**.76**	.77	1/280
SE-SS, T=1	.72	.74	.77	**60**
SE-SS, T=4	.73	.75	.77	30
SE-MS, T=4	**.74**	**.76**	**.78**	6

Table 1. Edge detection results on BSDS500 [1]. Our Structured Edge (SE) detector achieves top performance on BSDS while being 1-4 orders of magnitude faster than methods of comparable accuracy. Three variants of SE are shown utilizing either single (SS) or multiscale (MS) detection with variable number of evaluated trees T. SE-SS, $T = 4$ achieves nearly identical accuracy as gPb-owt-ucm [1] but is dramatically faster. [†Indicates results were measured on BSDS300; ‡indicates a GPU implementation.]

	ODS	OIS	AP	FPS
gPb [1] (rgb)	.51	.52	.37	1/240
SCG [31] (rgb)	.55	.57	.46	1/280
SE-SS (rgb)	.58	.59	.53	**30**
SE-MS (rgb)	**.60**	**.61**	**.56**	6
gPb [1] (depth)	.44	.46	.28	1/240
SCG [31] (depth)	.53	.54	.45	1/280
SE-SS (depth)	.57	.58	.54	**30**
SE-MS (depth)	**.58**	**.59**	**.57**	6
gPb [1] (rgbd)	.53	.54	.40	1/240
SCG [31] (rgbd)	.62	.63	.54	1/280
SE-SS (rgbd)	.62	.63	.59	**25**
SE-MS (rgbd)	**.64**	**.65**	**.63**	5

Table 2. Edge detection results on the NYU Depth dataset [33] for RGB-only (top), depth-only (middle), and RGBD (bottom). Across all modalities on all measures SE outperforms both gPb and SCG while running 3 orders of magnitude faster.

5. Results

In this section we show results on two different object contour datasets measuring both detection accuracy and runtime performance. We conclude by demonstrating the cross dataset generalization of our approach by testing on each dataset using decision forests learned on the other.

BSDS 500: We begin by testing on the popular Berkeley Segmentation Dataset and Benchmark (BSDS 500) [25, 1]. The dataset contains 200 training, 100 validation and 200 testing images. Each image has hand labeled ground truth contours. Edge detection accuracy is evaluated using three measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP) [1]. Prior to evaluation, we apply a standard non-maximal suppression technique to our edge maps to obtain thinned edges [5]. Example detections on BSDS are shown in Figure 2.

We evaluate our Structured Edge (SE) detector computed at a single scale (SS) and at multiple scales (MS). For SE-SS we show two results with $T = 1$ and $T = 4$ evaluated decision trees at each location. Precision/recall curves are shown in Figure 5 and results are summarized in Table 1. Our multiscale approach either ties or outperforms the state-of-the-art approaches [1, 31, 21], while being multiple orders of magnitude faster than [1, 31] and 6$\times$ faster than [21] (all frame rates are reported on an image size of 480×320 for all methods). With only minimal loss in accuracy, our single scale approach further improves the runtime by 5$\times$ to 10$\times$. In fact, with $T = 1$, we can perform at a frame rate of 60hz. This is considerably faster than [1, 31] while reducing the ODS score from 0.74 to 0.72. Note that the GPU implementation [6] of [1] only achieves an ODS score of 0.70 with a runtime of 2 seconds.

In comparison to other learning-based approaches to edge detection, we considerably outperform [9] which computes edges independently at each pixel given its surrounding image patch. We slightly outperform sketch tokens [21] in both accuracy and runtime performance. This may be the result of sketch tokens using a fixed set of classes for selecting split criterion at each node, whereas our structured forests can captured finer patch edge structure.

NYU dataset: The NYU Depth dataset (v2) [33] contains $1,449$ pairs of RGB and depth images with corresponding semantic segmentations. Ren and Bo [31] adopted the data for edge detection allowing for testing edge detectors using multiple modalities including RGB, depth, and RGBD. We use the exact experimental setup proposed by [31] using the same 60%/40% training/testing split (and use $1/3$ of the training data as a validation set) with the images reduced to 320×240 resolution (preprocessing scripts available from [31]). In [31] and our work, we treat the depth channel in the same manner as the other color channels. Specifically, we recompute the gradient channels over the depth channel (with identical parameters) resulting in 11 additional channels. Example SE results are shown in Figure 4.

In Table 2 we compare our approach to the state-of-the-art approaches gPb-owt-ucm (adopted to utilize depth) and SCG [31]. Precision/recall curves for all approaches are shown in Figure 3. Across all measures, our approaches (SE-SS and SE-MS) perform significantly better than SCG when using RGB only and depth only as an input. For RGBD our multi-scale approach performs considerably bet-

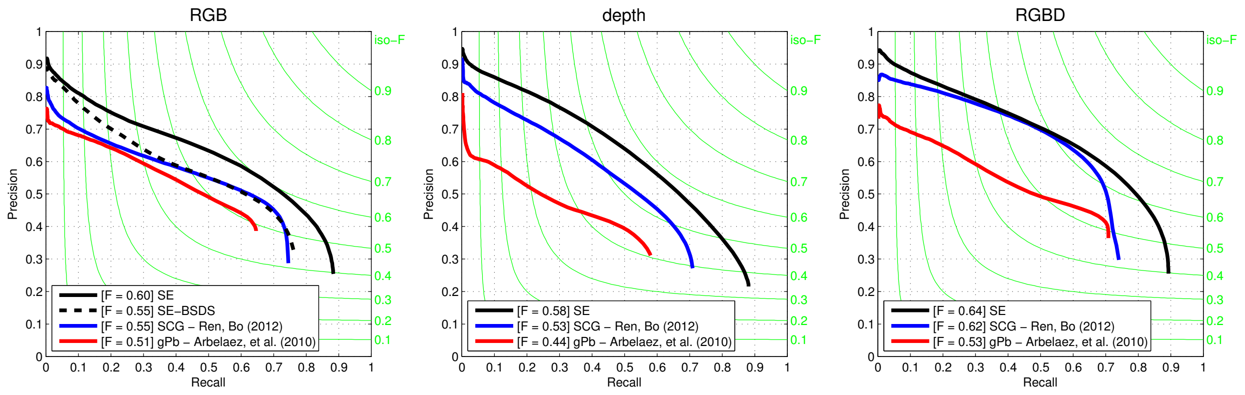

Figure 3. Precision/recall curves for the NYU Depth dataset using different image modalities. See Table 2 and text for details.

	ODS	OIS	AP	FPS
BSDS /BSDS	.74	.76	.78	6
NYU / BSDS	.72	.73	.76	6
BSDS / NYU	.55	.57	.46	6
NYU / NYU	.60	.61	.56	6

Table 3. Cross-dataset generalization for Structured Edges. TRAIN/TEST indicates the training/testing dataset used. Our approach exhibits strong cross-dataset generalization, a critical component for widespread applicability.

ter, while our single scale approach is similar to SCG. These improved scores are achieved while improving runtime performance by multiple orders of magnitude. In all cases both our approach and SCG outperform gPb by a wide margin indicating gPb may be overly tuned for the BSDS500 dataset.

Cross dataset generalization: To study the ability of our approach to generalize across datasets we ran a final set of experiments. In Table 3 we show results where we tested on NYU using structured forests trained on BSDS500 and tested on BSDS500 using structured forests trained on NYU. In both cases the scores remain high. In fact, when tested on the NYU dataset using BSDS500 as training, we achieve the same scores as SCG using NYU as training and significantly outperform gPb-owt-ucm (see also Figure 3). We believe this provides strong evidence that our approach could serve as a general purpose edge detector.

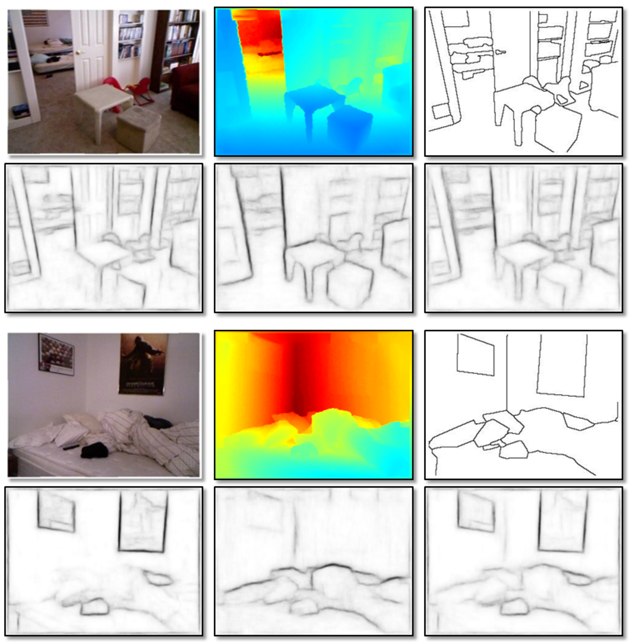

Figure 4. Illustration of two edge detection results on the NYU dataset: (top left) original image, (top middle) depth image, (top right) ground truth, (bottom left) SE-MS RGB only, (bottom middle) SE-MS depth only, and (bottom right) SE-MS RGBD.

6. Discussion

Our approach is capable of realtime frame rates while achieving state-of-the-art accuracy. This may enable new applications that require high-quality edge detection and efficiency. For instance, our approach may be well suited for video segmentation or for time sensitive object recognition tasks such as pedestrian detection.

Our approach to learning structured decision trees may be applied to a variety of problems. The fast and direct inference procedure is ideal for applications requiring computational efficiency. Given that many vision applications contain structured data, there is significant potential for structured forests in other applications.

In conclusion, we propose a structured learning approach to edge detection. We describe a general purpose method for learning structured random decision forest that robustly uses structured labels to select splits in the trees. We demonstrate state-of-the-art accuracies on two edge detection datasets, while being orders of magnitude faster than most competing state-of-the-art methods.

Source code will be made available online.

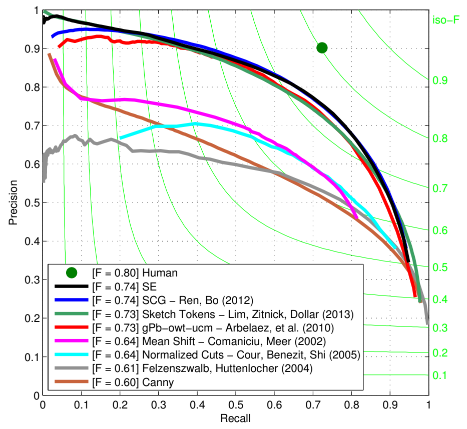

Figure 5. Results for BSDS 500. See Table 1 and text for details.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33, 2011. 1, 2, 4, 6

[2] M. Blaschko and C. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008. 2

[3] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical roc curves. *Computer Vision and Image Understanding*, 84(1):77–103, 2001. 2

[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. 2, 3

[5] J. Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, November 1986. 1, 2, 6

[6] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *ICCV*, 2009. 2, 6

[7] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, February 2012. 2, 3, 4

[8] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *BMVC*, 2010. 5

[9] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 1, 2, 6

[10] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973. 1, 2

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 6

[12] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 30(1):36–51, 2008. 1

[13] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE TOC*, 100(6), 1975. 1, 2

[14] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *PAMI*, 13:891–906, 1991. 1, 2

[15] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learn*, 63(1):3–42, Apr. 2006. 2, 3

[16] R. Hidayat and R. Green. Real-time texture boundary detection from ridges in the standard deviation space. In *BMVC*, 2009. 6

[17] T. K. Ho. The random subspace method for constructing decision forests. *PAMI*, 20(8):832–844, 1998. 3

[18] I. T. Joliffe. *Principal Component Analysis*. Springer-Verlag, 1986. 4

[19] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988. 1

[20] P. Kontschieder, S. Bulo, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *ICCV*, 2011. 1, 2, 3, 4

[21] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 1, 2, 4, 5, 6

[22] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *ECCV*, 2008. 2

[23] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 43, 2001. 1

[24] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004. 1, 2

[25] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 6

[26] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6:185–365, 2011. 1, 2

[27] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *PAMI*, 12(7):629–639, 1990. 2

[28] X. Ren. Multi-scale improves boundary detection in natural images. In *ICCV*, 2008. 5

[29] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using cond. random fields. In *ICCV*, 2005. 1

[30] X. Ren, C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, 2006. 1, 2

[31] X. Ren and B. Liefeng. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 1, 2, 5, 6

[32] G. S. Robinson. Color edge detection. *Optical Engineering*, 16(5), 1977. 1

[33] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *ICCV Workshop on 3D Representation and Recognition*, 2011. 2, 4, 6

[34] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *ICML*, 2005. 2

[35] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Learning for interdependent and structured output spaces. In *ICML*, 2004. 2

[36] S. Ullman and R. Basri. Recognition by linear combinations of models. *PAMI*, 13(10), 1991. 1

[37] S. Zheng, Z. Tu, and A. Yuille. Detecting object boundaries using low-, mid-, and high-level information. In *CVPR*, 2007. 1, 2

[38] D. Ziou, S. Tabbone, et al. Edge detection techniques-an overview. *Pattern Recognition and Image Analysis*, 8:537–559, 1998. 1, 2