Open access Journal International Journal of Emerging Trends in Science and Technology
Impact Factor: 2.838 DOI: http://dx.doi.org/10.18535/ijetst/v3i08.06

Fast Edge Detection Using Structured Forests

Authors
Bindu Nair[1], Ujwal Harode[2]
[1]B-404,Neelsidhi Atlantis, PIIT, Navi Mumbai, India
[2]Sector 19A,Nerul, New Panvel, Navi Mumbai, India
Email: *bindu_k_nair@hotmail.com, ujwal.harode@gmail.com*

ABSTRACT
An important and very crucial component of many systems involving images is the edge detection which includes object detectors and image segmentation algorithms. Edge patches always shows shapes of its inherent structure, like T junctions or lines. Taking advantage of the shapes available in the images we get to know of an edge detector which is accurate and efficient. In this approach the structured labels is robustly mapped to a space that is discrete by which evaluation of the measure of the standard information gain is possible. The outcome is a way that shows a real time performance which is many times faster than any of the traditional and modern approaches available today, which also achieves the best results for edge detection both the Berkeley Segmentation Dataset and Benchmark (BSDS 500) and the NYU depth dataset. This approach also shows its strength as a an edge detector that can be used for any regular purpose by displaying how this method generalize across all datasets
Keywords: *Segmentation, structured random forests, Principal component analysis, giniim purity, real time systems.*

INTRODUCTION

Edge detection can be said to a method by which the sharp change in image brightness or discontinuities can be identified using mathematical methods. The sharp change in image brightness are always depicted by a set of straight or curved lines at some points which is nothing but the edges. Image processing requires edge detection which is considered to be its main tool, same being applicable for machine and computer vision, especially in the those parts where we detect and extract features.

It has remained as an critical component in computer vision since the early days. It is also a critical pre-processing step for many important tasks, which includes object recognition, segmentation and active contours. Traditional approaches generally computes the colour gradients and then does a non-maximal suppression. But, many a times edges do not show any relation to the colour gradients like texture edges and imaginary contours. A number of features are taken as input in many method, which includes brightness, colour, depth gradients and the texture.

Since many edges shows relation to a number of visual characteristics, looking out for a single method to find the edges is not easy. Recent approaches proposes to take an image patch and then check whether the pixel located in the centre contains an edge. Optionally, then the edges which are predicted independently may be combined using global reasoning. Edges of a single patch will always be dependent on each other, which contains patterns like straight lines, parallel lines or junctions.

By this edge detection method, a generalized structured learning approach is taken which makes use of the available shapes in the patches of edge. This method can compute edges in real time, which is many times faster than most of the recent approaches. Here a method is used in which structured information is captured using the random forest framework. The function to split each of the branches in the tree is determined using structured labels. Each of the forest predicts the pixel labels in a patch of which an aggregate is taken to compute the final edge map.

Recent Approaches

Over the past 50 years many approaches has been suggested in the field of edge detection. Initial approaches concentrated more on intensity detection or colour gradients. The peak gradient magnitude which is in right angle to the edge detector is taken by the Canny edge detector. Recent approaches explore edge detection in images in which there are textures.

A lot of techniques have tried edge detection using learning. Boosted classifier was used in a method to separately label each pixel taking into account the surrounding image patch. Some used a method to combine the high, mid and low level cues which shows improvement for edge detections which are object specific. Recently, a method was introduced which made an improvement where it computed the gradients across learned sparse codes of patch gradients. The results were good, but it increased the already high cost of computation. Parallel algorithms was used to improve run time.

Then, an edge detector was used which used random forest classifiers to classify edge patches into sketch tokens, and also attempts to capture local edge structure. These sketch tokens are directly computed from patches of the colour image not from edge maps which were computed earlier. It gives a well planned approach towards edge detection which is also apt for object detection. Previously defined classes of edge patches is not required here.

For sequences, object pose, strings, graphs, object pose, bounding boxes etc where the space is arbitrarily complex in case of the input or output, it faces problem of learning the mapping which is addressed by structured learning.

Structured random forests is different from the rest in several respects. It is assumed that the output space operates on a standard input space and it is structured. This model will only output examples which are observed at the time of training, which assumes that a set of samples exist

This method was inspired by a recent paper on learning random forests for structured class labels for cases which are specific and the output labels represent a semantic image labelling for a particular image patch. It was observed that for a given colour image patch, any output can be stored at each leaf that it is not dependent on the structured semantic labels alone. Using the above knowledge, a general learning framework is considered for structured output forests which can be utilised for a broad class of output spaces and then apply he framework to learn about a fast and accurate edge detector.

Objective

For edge detection a structured forest formulation is applied. The input is a multiple channel image, maybe a RGB or RGBD image. Each pixel is labelled with a binary variable to indicate the presence of an edge or not.

In a given set segmented training images, the contours are the thresholds between the segments. For a given patch of image, this can be referred as a *segmentation mask* which indicates segment membership for all the pixel in the image or as a edge map in defined in binary terms.

The input features x, mapping function Π_{φ} which is used to determine splits, and the ensemble model which combines the multiple predictions are then computed.

RANDOM DECISION FORESTS

An example x2X is ordered by a choice tree ft(x) until the leaf hub by expanding in a recursive way to one side or directly down the tree. Every hub j in the tree is connected with a split capacity like

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

On the off chance that $h(x,\theta j) = 0$ hub jx bifurcates to one side, generally to one side and ends at the leaf hub. For info x the yield is the outcome came to by the tree at the leaf hub, which is either the division mark y 2Y or the dissemination over the labels Y.

A subjectively complex split capacity is h (x,θ).Another decision is a "stump" where x with a solitary element measurement is contrasted

with a limit. In particular, θ = (k,τ) and h(x,θ) = [x(k) < τ], where [·] signifies the marker capacity. Another prevalent decision is θ = (k1,k2,τ) and h(x,θ) = [x(k1) − x(k2) < τ]. Both are computationally productive and powerful practically speaking [7].

A group of T independent trees f_t makes a decision forest. Predictions $f_t(x)$ for an input x from the set of trees are merged into a one output with an *ensemble model*. Ensemble models are selected based on particular a problem and are dependent on Y, where classification uses majority voting and regression makes use of averaging, but there are even more enhanced ensemble models.

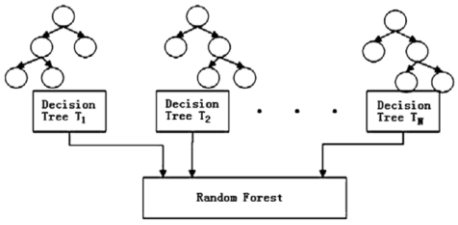


Fig 1: Random decision tree/forest

Any kind of information can be stored at the leaf node, but it depends only on the input *x*, and while the for castings from multiple trees has to be combined by the ensemble model, the leaf node stores any output *y*. Hence any complex outputs can be used which includes structured outputs.

Training the random decision forests with structured segments is difficult though the inputs are straight forward.

Training Decision Trees

Training of the trees is done in a recursive manner. The goal is to find the parameter $\theta_j$for a given node *j* and training set S_j which is nothing but the split function $h(x,\theta_j)$ which will result in a 'good' split of the data for which we need to define the *information gain criterion* :

$$Ij=I(Sj,SjL,SjR) \quad (2)$$

Where $^{S}_{j}{}^{L}$ = {(*x*,*y*) ∈S_j|$h(x,\theta_j)$ = 0}, $^{S}_{j}{}^{R}$ =.$S_j \backslash ^{S}_{j}{}^{L}$

Maximum value of the information gain I_j gives the splitting parameter θ_j. Training then continues recursively towards the left node with data$\mathcal{S}_j^L$and similarly for the right node. It stops when a maximum depth is achieved or the information gain or training set size goes below the decided threshold values.

The standard definition of information gain for multiclass classification is :

$$I_j = H(\mathcal{S}_j) - \sum_{k\in\{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k) \quad (3)$$
$$H(S) = \frac{1}{|\mathcal{S}|}\sum_y (y-\mu)^2$$
$$\mu = \frac{1}{|\mathcal{S}|}\sum_y y$$

$H(S) = p_y \log(p_y)$ denotes the Shannon entropy and p_y is the fraction of elements in S with label *y*. Alternatively the Gini impurity $H(S) = p_y(1-p_y)$ has also been used in conjunction with Eqn. (3) [5].

For regression, entropy and information gain can be extended to continuous variables[6]. Alternatively, a common approach for single-variate regression (Y = R) is to minimize the variance of labels at the leaves [4]. If we write the variance as where, then substituting *H* for entropy in Eqn. (3) leads to the standard criterion for single-variant regression.

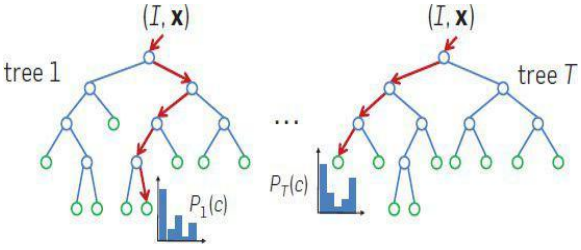


Fig 2 : Training decision trees

Randomness and optimality

Individual decision trees exhibit high variance and tend to over-fit [5,9]. 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.

High variance and over fitting is exhibited by individual decision trees [5,9] ,hence output from a number of de-correlated trees are used by the

decision forests to improve the results. Achieving a sufficient

High accuracy models can be achieved by injecting randomness at the level of nodes and has proven to be 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 $\theta = (\sqrt{k},\tau)$ and $h(x,\theta) = [x(k) < \tau]$, [9] advocates sampling d features where $X = R^d$ and a single threshold τ per feature.

To achieve this high diversity ensemble the accuracy of individual trees has to be sacrificed [9].

STRUCTURED RANDOM FORESTS

Random decision forests is applied to general structured output spaces Y. If $x \in X$ represents an image patch and $y \in Y$ represents the corresponding local image annotation maybe a segmentation mask or set of semantic image labels.

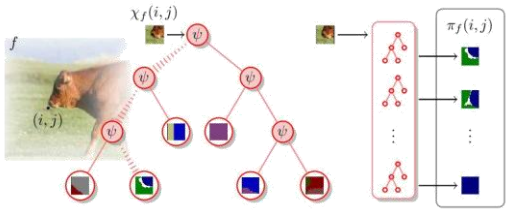


Fig 3 : Structured random forest

There are two main challenges faced by the structured labels when trained by the random forests which is high dimensionality and also being complex. Therefore the selected candidate splits over structured labels may prove to be expensive. More critically the information gain over these structured labels may not be well defined.

But as per observation even an *approximate* value of information gain is enough to train effective random forest classifiers. Exact values are not required. So the basic plan is to map all the structured labels $y \in Y$ at a given node into a discrete set of labels $c \in C$, where $C = \{1,...,k\}$, such that *similar* structured labels y are assigned to the same discrete label c.

Now the information gain calculated directly and efficiently over C can give the same results for the information gain if calculated over the structured labels Y. By this method for each node we can make use of the existing random forest training procedures to learn structured random forests effectively.

Calculating information gain relies on measuring similarity over Y. But for edge detection, calculating the similarity over Y is not defined properly. So here Y is mapped to an intermediate space Z where measuring of distance is easier. Hence a two stage approach is used where Y is mapped first to Z followed the Z being mapped to the discrete space C.

Intermediate Mapping Π

Here mapping is done in the form:

$$\Pi : Y \rightarrow Z$$

So that the an approximate value of the dissimilarity of $y \in Y$ can be taken by computing the Euclidean distance in Z. Here the labels yY are 16×16 segmentation masks and $z = \Pi(y)$ is defined as a long binary vector which encodes whether every pair of pixels in y belong to the same or different segments. It is easier to measure the distance in Z.

But computation of Z is also a challenge as it may be high dimensional. Like for edge detection there are $\binom{16\cdot16}{2} = 32640$ unique pixel pairs in a segmentation mask of 16 × 16 size, so computing z for every y would be expensive. Since only an approximate distance value is required, the dimensions of Z can be brought down.

$$\mathcal{Y} \rightarrow \mathcal{Z} \rightarrow \mathcal{C}$$

Structured Space — Intermediate space — discrete space

Fig 4 : Intermediate mapping

For this, m dimensions of Z is sampled, which will result in a reduced mapping $\Pi_\varphi : Y \rightarrow Z$ parameterized by φ. While training, the training labels Y_j at each node is applied with a distinct mapping Π_φ which is randomly generated. That serves two requirements, first, Π_φ can be

considerably faster Π. Second, taking samples of Z will also make it more random to the learning process and helps in getting more diversity of trees.

In the end, Principal Component Analysis (PCA) is also used to reduce the dimensions of Z even further. This also helps in reducing the noise while giving an approximate Euclidean distance. In practice, Π_φ with m = 256 dimensions followed by a PCA projection to at most 5 dimension is used.

Information Gain Criterion

A number of choices are possible for the information gain criterion using the mapping Π_φ : Y → Z. For discrete Z multi-variant joint entropy could be computed directly. Here it is observed that $m \geq 64$ is required to get an accurate value of similarities between the various Z values. Also if a continuous Z is given, variance or a continuous formulation of entropy [7] can be used to define information gain. Here a simple and more efficient method is used.

A set of structured labels $y \in$ Y is mapped into a discrete set of labels $c \in$ C, where C = {1,...,k}, in a way that labels z with same properties are designated to the same discrete label c. These labels may be binary (k = 2) or multiclass (k > 2) hence standard information gain criteria based on Shannon entropy or Gini impurity as given in Eqn. (3) can be used. This method of discretization is performed while training each node and is dependent on the distribution of labels at a given node.

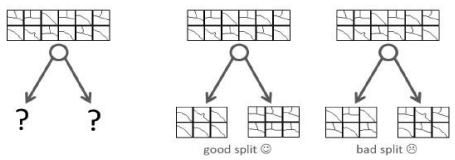


Fig 5 : Intermediate node splits

There are two ways by which this method can be applied. The first method is to cluster z into k clusters using K-means or by second method which quantize z based on the top $\log_2(k)$ PCA dimensions, giving z a discrete label c according to the orthant (generalization of quadrant) into which z falls. Both methods perform almost in the same manner but quantization is slightly faster.

PCA quantization with k = 2 is used.

Ensemble Model

Here it is defined how to combine a set of n labels $y_1...y_n \in$ Y into a single prediction for both stages that is training (to associate labels with nodes) and testing (to merge multiple predictions). As earlier, take a sample of Π_φ with m dimensions and then $z_i = \Pi_\varphi(y_i)$ for each i. Choose the label y_k whose z_k is the medoid (z_k that minimizes the sum of distances to all other z_i[1]).

This ensemble model depends on m and the selected mapping Π_φ. However, medoidis computed for only small n, hence only a coarse distance metric is enough to select a representative element y .

The only drawback is that any prediction $y \in$ Y must have been seen at the time of training, the ensemble model will not synthesize new labels. So you need additional information about Y. But in practice, domain specific ensemble models can be used. For edge detection the default ensemble model is used for training but utilize a tailor made method for merging all the outputs over a number of overlapping image patches

EDGE DETECTION USING STRUCTURED FORESTS

For edge detection a structured forest formulation is applied. The input is a multiple channel image, maybe a RGB or RGBD image. Each pixel is labelled with a binary variable to indicate the presence of an edge or not.

In a small image patch the labels are always highly interdependent, which provides a promising candidate problem for the structured forest approach.

If there is a given set of segmented training images, in which the boundaries between the segments correspond to contours[2]. From an image patch, its representation can be described either as a *segmentation mask* indicating segment

membership for each pixel or a binary *edge map*. Use $y \in Y = Z^{d \times d}$ to denote the mask and $y0 \in Y0 = \{0,1\}$ for the edge map, d in this case denotes the patch width. The edge map y can always be formulated from the segmentation mask y, but not vice versa. Both representations are used here We then 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

Using a 32 x 32 patch of image, a 16 × 16 structured segmentation mask is predicted. Then enhance each of the image patch with multiple additional *channels* of information, which will give a result vector
$R^{32 \times 32 \times K}$where K is the number of channels.
Pixel lookups $x(i,j,k)$ and pair wise differences $x(i_1,j_1,k) - x(i_2,j_2,k)$ both features are used.
In the CIE-LUV colour space, almost identical set of colour and gradient channels are used to compute three colour channels together with the normalized gradient magnitude at two scales i.e full and half resolution. On the basis of its orientation, divide each gradient magnitude channel into four channels. A triangle filter with radius 2 is used to blur these channels and then sampled down by a factor of 2. We get a three colour, two magnitude and eight orientation channels, which makes thirteen channels in all.
The channels are then again down sampled by a factor of 2, so that we get 32·32·13/4 = 3328 candidate features $x(i,j,k)$. Pair wise difference features is also calculated. An eight pixel radius large triangle blur is applied to each of the channels, and then down sample to 5 × 5 resolution. When all pairs of candidates are sampled and computed, the difference between them gives an extra $\binom{5 \cdot 5}{2} = 300$ candidate features for every channel, which gives a total of 7228 candidate features for every patch.

Mapping Function

For training the decision trees, a mapping function Π: Y → Z is defined. The 16x16 segmentation masks are the structured labels y.
Either use Π: Y → Y^0, where y^0 is the binary edge map equivalent to y but the Euclidean distance over Y^0 gives a weak distance measure.
So a substitute mapping Π is defined, where if $y(j)$ for $1 \le j \le 256$ denotes the j^{th} pixel of the mask(y) but a single value $y(j)$ gives no details about y since it is defined only up to a permutation. But if a pair of locations j_1 6= j_2 is sampled and then check if $y(j_1) = y(j_2)$, this defines $z = \Pi(y)$ as a large binary vector that encodes $[y(j_1) = y(j_2)]$ for every pair of indices j_1 6= j_2 that is unique. Since Z has $\binom{256}{2}$dimensions, only a subset of m dimensions is computed .If we set $m = 256$ and $k = 2$ the similarity of segmentation masks is effectively captured.

Ensemble Model

Random forests achieves strong results while the outputs of multiple de-correlated trees are combined. Averaging is possible with multiple edge maps y^0 $2Y^0$ to give a soft edge response though merging of multiple segmentation masks y 2Y is difficult. Since a decision tree is capable of storing arbitrary information at its leaf nodes, we can also store the almost similar edge map y^0 along with the learned segmentation mask y. Averaging can then be done on the predictions obtained from multiple trees. The efficient use of structured labels that captures information for an entire image neighbourhood is reduced has resulted in reduced number of decision trees T to be evaluated for every pixel. Thus with a 16×16 output patch, each pixel receives $16^2T/4 \approx 64T$ predictions, though in practice $1 \le T \le 4$ is used which is possible for an image with a step of two pixels because the structured output is computed heavily on it.
As the inputs and outputs of all trees overlap, training a total of $2T$ trees and then evaluating an alternate set of T trees at every adjoining location helps. By putting in a larger gap between the trees improves results to a certain extent but will not give any further improvement.

ENHANCEMENT TECHNIQUES

Multi scale Detection (SE+MS)

For a given input image, the detector is run on all the resolution versions, half, original and double of the original image and then the results from all three are averaged after the image is got back to its original dimensions. This approach improves the edge quality.

Edge Sharpening (SE+SH):

Non maximal suppression is used on the edges predicted from this method since there are not clear and this can help in detecting strong, isolated edges. But for fine image structures the edges in the image may blend together which will result in missed detections whereas for edges which are not strong no detection may happen. Diffused edge responses are due to the edge maps which are individually predicted where the edges are already noisy and not very perfectly aligned to the image data under it or to each other.

It is a sharpening procedure in which edge responses from overlapping predictions are aligned. To exactly localize the predicted responses, values of the local image colour and depth values can be used. The predicted segmentation mask if morphed slightly is better matched with the underlying image. A sharper and better localized edge responses can be achieved by aligning the underlying image data and the masks over it.

Now compute and average the edge maps derived by using the new sharpened segmentation masks. Butthe resulting edge map is sharper, since the edge maps are better aligned to the image data. Repeat the procedure multiple times before taking an average of the corresponding edge maps. In practice only two steps are required where initial sharpening step gives the best results.

Sharpened segmentation masks, compute and average their corresponding edge maps as before. However, since the edge maps are better aligned to the image data the resulting aggregated edge map is sharper. Sharpening can be repeated multiple times prior to averaging the corresponding edge maps. Experiments reveal that the first sharpening step produces the largest gains, and in practice two steps suffice. Taking advantage of the sparcity of edges, the sharpening procedure can be implemented efficiently.

Across all modalities on all measures SE outperforms both gPb and SCG while running 3 orders of magnitude faster.

In Table 6.2 this method is compared to the state-of-the-art approaches gPb-owt-ucm (adopted to utilize depth) and SCG [16]. When using RGB and depth individually as an input, SE-SS and SE-MS perform significantly better than SCG. For RGBD the multi scale approach performs considerably better, while the single scale approach is similar to SCG. The improved scores are also accompanied by a runtime which is orders of magnitude faster than the rest.

PARAMETER TESTING

Here the performance of the structured edge detector is verified in detail. First the influence of parameters is verified in Section 6.1 and then test the SE variants in Section 6.2. Then compare the results on the BSDS [1] and NYUD [44] datasets to the other recent approaches with respect to both accuracy and runtime. Also verified is the cross dataset generalization of this approach.

The majority of the testings are performed on the Berkeley Segmentation Dataset and Benchmark (BSDS500) [1], [35]. The dataset contains 200 training, 100 validation, and 200 testing images. Each image has hand labeled ground truth contours. Three standard measures Optimal Dataset Scale (ODS) or fixed contour threshold, Optimal Image Scale (OIS)per-image best threshold and average precision (AP) is used to test the edge detection accuracy . Recall at 50 per-cent precision (R50) is used to evaluate accuracy in the high recall regime. A standard non-maximal suppression is used before evaluation to the edge maps to obtain thinned edges[7].

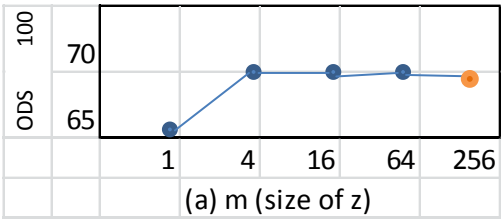
Parameter sweeps

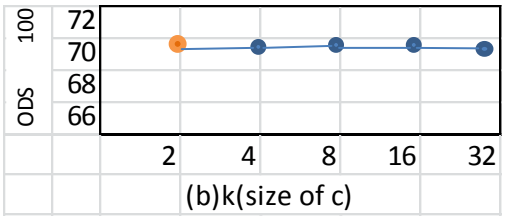
BSDS validation set is used to set all parameters which is completely independent of the test set. Parameters include: structured forest splitting parameters (e.g., m and k), feature parameters (e.g., image and channel blurring), and model and tree parameters (e.g. number of trees and data quantity). Training takes 20 minute per tree using one million patches and is parallelized over trees. All trees are evaluated in parallel. Quad-core machine is used for all reported runtimes.

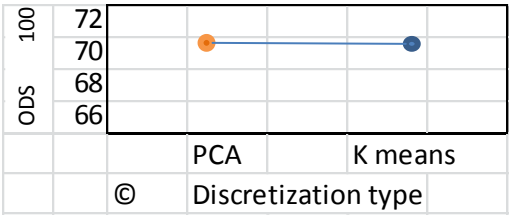
The effect of choices of splitting, model and feature parameters are verified. Training is done on the 200 image training set and then verify edge detection accuracy on the 100 image validation set. An average of the results are done after five trials. Initially all parameters are set to their default values indicated by orange markers in the plots. Then, the parameters are changed with only one parameter kept fixed to check the effect on the edge detection accuracy.

Training is done using fewer patches and utilize only sharpening (SH) not the multi scale detection (MS). Since the validation set is more challenging than the test set evaluation is done using 25 thresholds instead of 99 further reducing accuracy (.71 ODS).

Splitting Parameters : How best to measure information gain over structured labels is verified in fig 5 Plots (a) and (b) shows that m should be large and k small. Results are robust to both the discretization method and the discrete measure of information gain as shown in plots (c) and (d).







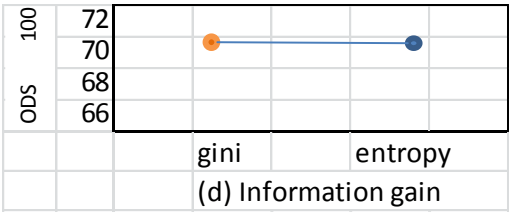
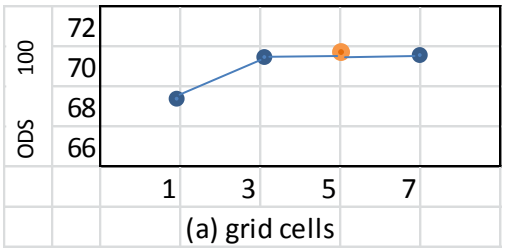
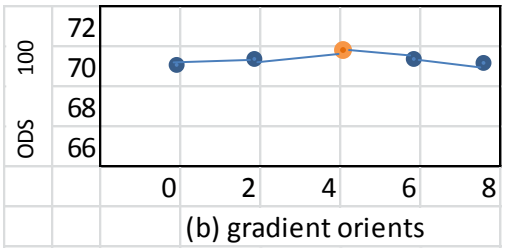
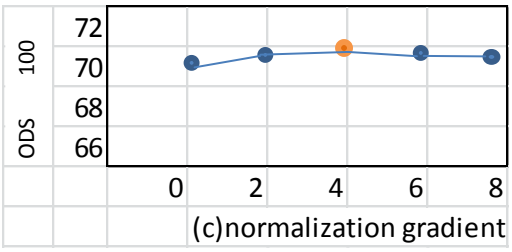


Fig 6 : Splitting parameters sweep

Feature Parameters: How varying the channel features affects accuracy is shown in fig 6. Here it is seen that performance is relatively insensitive to a broad range of parameter settings.







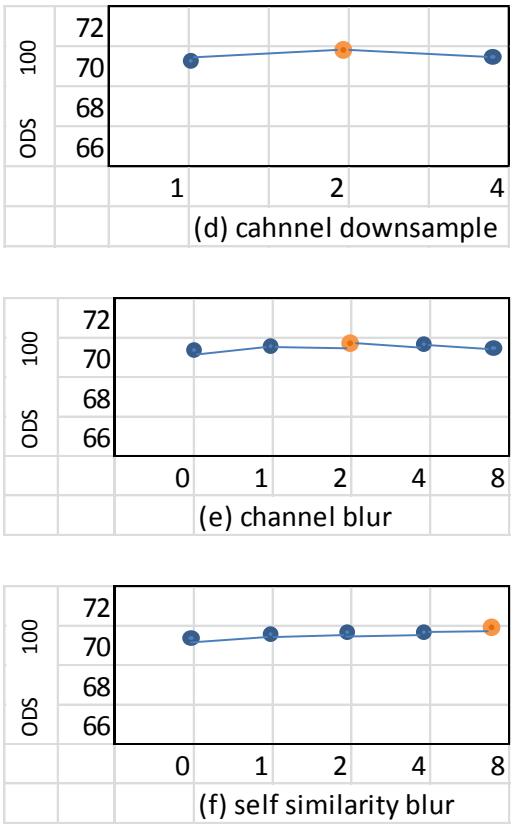
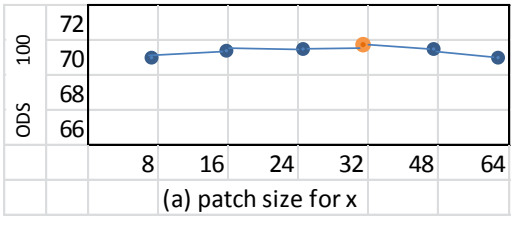
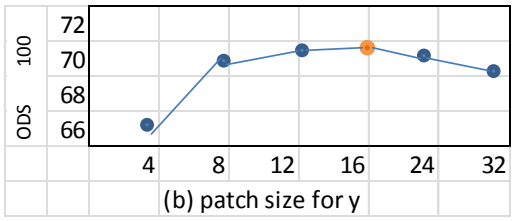
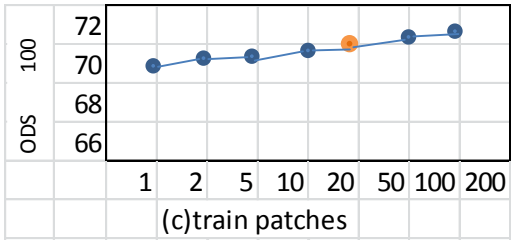


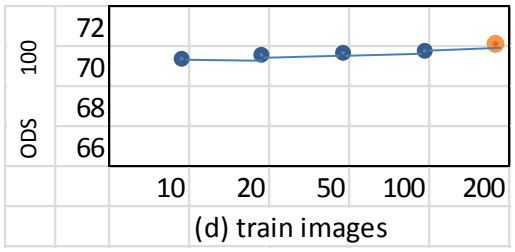
Fig7 : Features parameter sweep

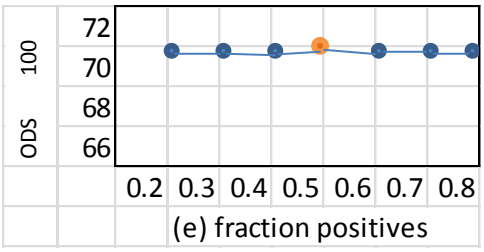
Model Parameters: The influence of parameters governing the model and training data are plotted in Fig 7. The effect of image and label patch sizes on accuracy is shown in (a) and (b), 32x32 image patches and 16x16 label patches are seen to be the best choice. More number of patches and training images improves the accuracy as shown by (c) and (d).(e) shows that about half the sampled patches should be 'positive' (have more than one ground truth segment) and (f) shows that training each tree with a fraction of proportionally lower memory usage). In (g)-(i) we see that many, deep, un-pruned trees give best performance (nevertheless, we prune trees so every node has at least eight training samples to decrease model size). Finally (j) shows that two sharpening steps give best results.

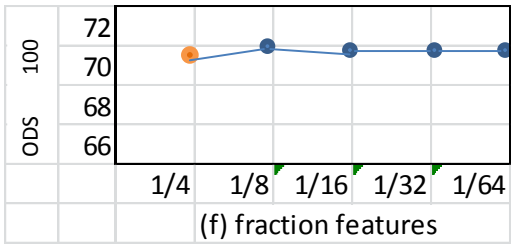


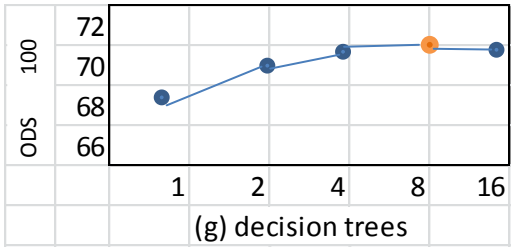


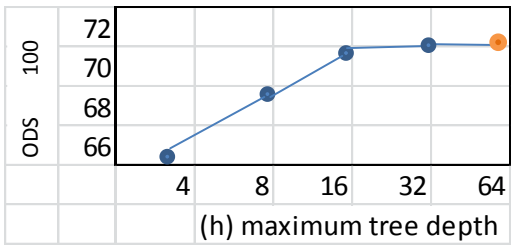


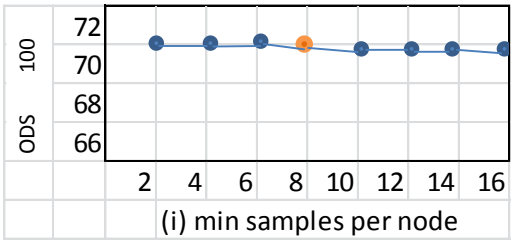












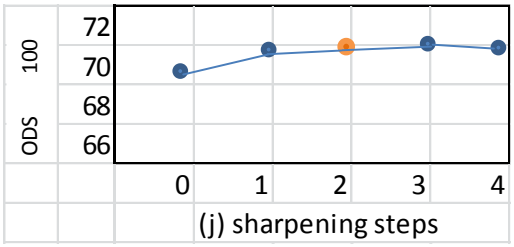


Fig 8 : Model parameter sweep

Structured Edge variants

Edge sharpening (SH) and multi scale detection (MS) is used to give more accurate results. The performance of the four different combinations are analysed here. The model which has been trained can also be used for both sharpening and multi scale detection SE, SE+MS, SE+SH, and SE+MS+SH are the four combinations for which the precision /recall curve is plotted. Summary for all four has been given in the bottom rows of Table 6.1. SE has an ODS score of .73 and SE+MS and SE+SH both has an ODS score of .74. SE+MS+SH, which is a combination of multi scale detection and sharpening gives an ODS of .75. For all the combinations OIS is two points higher than ODS which is measured using a separate optimal threshold per image.

Accuracy is improved by both sharpening and multi scale detection. R50 measures accuracy in the high recall regime. R50 for SE is .90 but SE+MS does not make any improvement to this. Whereas, SE+SH increases R50 to .93. Multi scale makes a considerable improvement to the precision in the low-recall regime. On an average, both SH and MS improves AP, where SE+SH+MS achieves an AP of .80.

The runtime for the four combinations are shown in the last column of Table 6. 1.Real time processing is enabled by SE that runs at a frame rate of 30 hz. MS incurs high cost as both SH and MS slow the detector. SE+SH achieves good accuracy as it run sat over 12 hz. So SE + SH achieves best results in the high recall regime needed for most common requirements so for see it as the default variant for various applications.

BSDS 500 Results: Testing is done on the Berkeley Segmentation Dataset and Benchmark (BSDS 500) which contains 200 training, 100 validation and 200 testing images. Each of the images has hand labelled ground truth contours. Three standard measures Optimal Dataset scale (ODS) also called the fixed contour threshold, Optimal Image scale (OIS) also called the per image best threshold and average precision (AP)is used to test the edge detection [2]. R50 that is recall at 50% precision is additionally used to evaluate accuracy in the high recall regime. Non-maximal suppression technique is used before evaluation to the edge maps to obtain thinned edges [19].

Detections examples on BSDS are shown in Fig 8

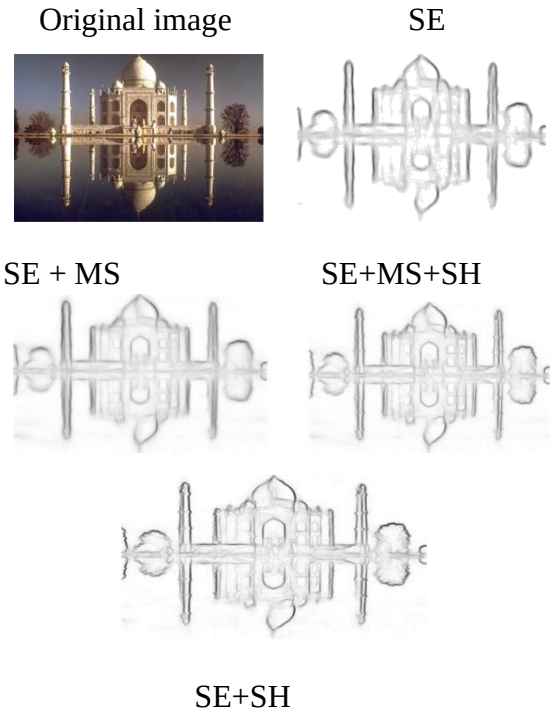


Fig 9 : Detection using BSDS dataset

The Structured Edge (SE) detector is tested by computing at a single scale (SS) and at multiple scales (MS). At T=1 and T=4 decision trees there are two results for SE-SS at each location. The state-of-the-art approaches are out per formed in speed by the multi scale approach. Single scale approach also betters the runtime by 5 to 10. With T = 1, the method can perform at a frame rate of 60hz which is faster than [2,19] while also bringing the ODS score from 0.74 to 0.72.

This method shows best performance on the BSDS while also being magnitudes of order faster than other methods that are equally accurate. Three variants of the Structured Edge detector are shown using either the single (SS) or multiscale (MS) detection with different numbers of evaluated trees T .For SE-SS, T = 4 achieves nearly identical accuracy as gPb-owt-ucm [1] but is also faster. Compared to other methods of similar approach to edge detection, this method considerably outperform[7] which computes edges independently at each pixel given its surrounding image patch. It also slightly outperforms sketch tokens [8] in both accuracy and runtime performance. This may be because the sketch tokens use a fixed set of classes for selecting split criterion at each node, whereas the structured forests captures finer patch edge structure.

Table 1: Edge detection results on BSDS

	ODS	OIS	AP	FPS
Human	0.8	0.8	-	-
Canny	0.6	0.64	0.58	15
Felz-Hutt [11]	0.61	0.64	0.56	10
Hidayat-Green [19]	.62y	-	-	20
BEL [9]	.66y	-	-	1/10
gPb + GPU [6]	.70y	-	-	1/2z
gPb [2]	0.71	0.74	0.65	1/240
gPb-owt-ucm [2]	0.73	0.76	0.73	1/240
Sketch tokens [9]	0.73	0.75	0.78	1
SCG [17]	0.74	0.76	0.77	1/280
SE-SS, T =1	0.72	0.74	0.77	60
SE-SS, T =4	0.73	0.75	0.77	30
SE-MS, T =4	0.74	0.76	0.78	6

NYU dataset Results: The NYU Depth dataset (v2) [18] contains 1,449 pairs of RGB and depth images with corresponding semantic segmentations. 60%/40% training/testing split is used and 1/3 of the training set is used for validation with the images reduced to 320 x 240 .The depth channel is treated similar to the other colour channels. Specifically, the gradient channels are re-computed over the depth channel (with identical parameters) resulting in 11 additional channels.

Table 2 : Edge detection results on NYUD

	ODS	OIS	AP	FPS
gPb [1] (rgb)	0.51	0.52	0.37	1/240
SCG[17] (rgb)	0.55	0.57	0.46	1/280
SE-SS (rgb)	0.58	0.59	0.53	30
SE-MS (rgb)	0.6	0.61	0.56	6
gPb[2] (depth)	0.44	0.46	0.28	1/240
SCG[17] (depth)	0.53	0.54	0.45	1/280
SE-SS (depth)	0.57	0.58	0.54	30
SE-MS (depth)	0.58	0.59	0.57	6
gPb [1] (rgbd)	0.53	0.54	0.4	1/240
SCG[17] (rgbd)	0.62	0.63	0.54	1/280
SE-SS (rgbd)	0.62	0.63	0.59	25
SE-MS (rgbd)	0.64	0.65	0.63	5

Cross dataset generalization: Table 6.3 shows results where we tested on NYU using structured forests trained on BSDS500 and tested on BSDS500 using structured forests trained on NYU. Both methods achieve a high score. When tested on the NYU dataset using BSDS500 as training, the scores achieved is same as SCG and using NYU as training outperforms gPb-owt-ucm. Therefore this method can serve general purpose edge detector.

Table 3: Edge detection results using cross dataset

	ODS	OIS	AP	FPS
BSDS /BSDS	0.74	0.76	0.78	6
NYU / BSDS	0.72	0.73	0.76	6
BSDS / NYU	0.55	0.57	0.46	6
NYU / NYU	0.6	0.61	0.56	6

CONCLUSION

This approach is capable of real time 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.

This 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, a structured learning approach to edge detection is proposed. A general purpose method for learning structured random decision forest that robustly uses structured labels to select splits in the trees is described. State of the art accuracies on two edge detection datasets, while being orders of magnitude faster than most competing state-of-the-art methods is described.

REFERENCES

1. Piotr Dollar and C. Lawrence Zitnick, "Fast Edge Detection Using Structured Forests," IEEE Trans. Pattern Anal. Mach. Intell, vol. 37, No. 8, Aug. 2015
2. P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 5, pp. 898–916, May 2011.
3. M. Blaschko and C. Lampert, "Learning to localize objects with structured output regression," presented at the European Conf. Computer Vision, Marseilles, France, 2008
4. K. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical ROC curves," Comput. Vis. Image Understanding, vol. 84, no.1, pp. 77–103, 2001.
5. L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp 5–32, Oct. 2001.
6. J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986. [7]
7. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis,vol. 3. New York, NY, USA: Wiley, 1973.
8. V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 30,no. 1, pp. 36–51, Jan. 2008.
9. S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2013, pp. 564–571.
10. P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees", Machine Learn, 63(1):3–42, Apr. 2006
11. P. Kontschieder, S. Bulo, H. Bischof, and M. Pelillo, " Structured class-labels in random forests for semantic image labelling", In ICCV, 2011
12. N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor", In ICCV Workshop on 3D Representation and Recognition, 2011
13. 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
14. B. Catanzaro, B.-Y.Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In ICCV, 2009
15. P. Dollar,´ Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In CVPR, 2006
16. X. Ren and B. Liefeng. Discriminatively trained sparse code gradients for contour detection. In NIPS, 2012

17. J. Lim, C. L. Zitnick, and P. Dollar´. Sketch tokens: A learned mid-level representation for contour and object detection. In CVPR, 2013
18. 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
19. P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient graph-base dimage segmentation," Int.J. Comput. Vis.,vol. 59,no.2,pp.167–181,200